(12)

NOSC TD 649

NOSC TD 649

Technical Document 649

COMPUTING CONFIDENCES IN TACTICAL RULE-BASED SYSTEMS BY USING DEMPSTER-SHAFER THEORY

R. A. Dillard

14 September 1983

Prepared for
NOSC Independent Research/
Independent Exploratory
Development (IR/IED) Program

Approved for public release; distribution unlimited.

NOSC

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

J.M. PATTON, CAPT, USN
Commander

R.M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This report summarizes work performed during fiscal year 1983 by a member of the C^2 Information Processing Technology Branch (NOSC Code 8233). The effort was supported by the NOSC Independent Research/Independent Exploratory Development (IR/IED) Program. Note that the software for the ROSIE program was licensed from the Rand Corporation, of which ROSIE is a registered trademark.

Released by
G.P. Allgaier, Head
Advanced Command Center
Technology Division

Under authority of
J.H. Maynard, Head
Command Control-Electronic
Warfare Systems and
Technology Department

RH

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NOSC TD 649	2. GOVT ACCESSION NO. AD-A137 274	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMPUTING CONFIDENCES IN TACTICAL RULE-BASED SYSTEMS BY USING DEMPSTER-SHAFER THEORY		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. A. Dillard		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ocean Systems Center San Diego, CA 92152		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NOSC Independent Research/ Independent Exploratory Development (IR/IED) Program
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Ocean Systems Center San Diego, CA 92152		12. REPORT DATE 14 September 1983 13. NUMBER OF PAGES 92
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial intelligence
Expert systems
Confidence weighing
Rule-based systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the implementation of confidence computing methods based on Dempster-Shafer Theory. The theory is applicable to tactical decision problems that can be formulated in terms of sets of mutually exclusive and exhaustive propositions. Dempster's combining procedure, a generalization of Bayesian inference, is used to combine probability mass assignments supplied by independent bodies of evidence. The computing procedures are implemented in the rule-based system ROSIE, and apply to all valid mass assignments. An ordering strategy is used to combine various kinds of assignments by using different procedures that exploit the special features of each. Applications to platform typing and contact association are demonstrated.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

1. INTRODUCTION

The work reported here builds on the first year's results (ref 1, 2) of an IR project aimed at designing confidence computing methods for rule-based systems. Early in this project, we chose to address only problems expressible as sets of exclusive hypotheses, since mathematical approaches for exclusive hypotheses are relatively tractable and since the immediate problems of concern can be expressed in that form. We found that we could arrange most of the evidence bearing on each problem in such a way that the resulting bodies of evidence are mutually independent. With the requirements of exclusive hypotheses and independent evidence met, Dempster's combining procedures (ref 3) and Shafer's representation scheme (ref 4) are applicable.

In the first year, we implemented Dempster's rule for the most common kinds of probability assignments derived from evidence and conducted experiments in two rule-based inference systems for propositions concerning ship type. The major achievements this year have been (a) implementation of Dempster's rule of combination for all valid probability assignments; (b) implementation of a contact association application; and (c) improved user interface mechanisms. The implementation this year has been in ROSIE (Rule-Oriented System for Implementating Expertise), a development of the Rand Corporation (ref 5).

2. The DEMPSTER-SHAFER THEORY

The simpler aspects of Dempster-Shafer theory are summarized here. Further descriptions can be found in references 2, 3, 4, and 6.

2.1 PROBABILITY MASS ASSIGNMENTS

Assume we have a set of n mutually exclusive and exhaustive propositions, labeled A1, A2, ..., An. Suppose there are several independent categories of evidence concerning these propositions, and an expert's interpretation of each contributes a "probability mass assignment" over the n propositions. While a Bayesian probability mass assignment would consist of n probabilities summing to unity, here we permit a more general assignment. In addition to permitting probability mass to be assigned to the original n propositions, we permit it to be assigned to disjunctions of propositions. There are $2^n - 1$ such general propositions that may be assigned mass, and the masses summed over all of these propositions must equal unity.

One important general proposition is the disjunction of all n original propositions,

$$\theta = A1 \vee A2 \vee \ldots \vee An,$$

where "v" denotes the Boolean OR. The mass assigned to θ represents mass assigned to uncertainty; it represents the uncertainty the expert has concerning the accuracy and the interpretation of the evidence.

To illustrate the concept of a probability mass distribution we consider the case of three exclusive and exhaustive propositions A1, A2, and A3. The general propositions are then

$$\begin{aligned}
B1 &= A1 = \tilde{}(A2 \vee A3) \\
B2 &= A2 = \tilde{}(A1 \vee A3) \\
B3 &= A3 = \tilde{}(A1 \vee A2) \\
B4 &= A1 \vee A2 = \tilde{}A3 \\
B5 &= A1 \vee A3 = \tilde{}A2 \\
B6 &= A2 \vee A3 = \tilde{}A1 \\
B7 &= A1 \vee A2 \vee A3 = \theta
\end{aligned}$$

where ~ denotes the Boolean NOT. Letting m(Bk) denote the probability mass assigned to proposition Bk, we have the constraint m(B1) + m(B2) + ... + m(B7) = 1. Letting mj(Bk) denote the mass assigned to Bk based on the jth category of evidence, we might typically have the following assignments from two categories:

Assignment 1: m1(A1) = .2, m1(A2) = .4, m1(A3) = .05, m(θ) = .35

Assignment 2: m2(~A2) = .4, m2(θ) = .6.

As described in the next section, probability mass assignments contributed by independent categories of evidence can be combined by Dempster's rule of combination into a single probability mass assignment.

The "support" for a general proposition Bk is the sum of the probability masses (from a single assignment, usually an assignment created by combining all previous assignments) contributed to all propositions Bh such that Bh&Bk = Bk, where the ampersand denotes the Boolean AND. For $n \geq 3$, for example, the support for the general proposition A1vA2vA3 is

$$s(A1vA2vA3) = m(A1) + m(A2) + m(A3) + m(A1vA2) + m(A1vA3) + m(A2\&A3) + m(A1vA2vA3).$$

From the definition of support and the fact that the masses must sum to unity, we have s(θ) = 1. From the definition of support we also see that the support for an original proposition Ai is simply the mass assigned to it; i.e., s(ai) = m(Ai). An important measure of the evidence against one of the original propositions is the support for its negation, s(~Ai). This measure is usually presented in its complement form

$$p(Ai) = 1 - s(\tilde{}Ai),$$

where p(Ai) represents the "plausibility" of Ai. For n = 3, for example, the plausibility of proposition A2 is

$$p(A2) = 1 - s(\bar{A}2) = 1 - s(A1 \vee A3)$$
$$= 1 - m(A1) - m(A3) - m(A1 \vee A3)$$
$$= m(A2) + m(A1 \vee A2) + m(A2 \vee A3) + m(\theta).$$

The capability of withdrawing or replacing a mass assignment is desirable when earlier information is found to be erroneous or when later related evidence supports a more accurate assignment. An example of the latter is when a contact's speed is measured over a period of time and an earlier assignment based on the initial measurement can be replaced with one reflecting a higher speed capability. As there is no satisfactory way of combining assignments from dependent bodies of evidence, an assignment should be based on the combined evidence. (The fact that the speed changed could be considered sufficiently independent for the purpose of contributing another assignment suggesting that the contact is not a merchant.) One step in implementing withdrawal mechanisms is to give each mass-assigning inference rule a tag that is a number or code name. An additional advantage of tagging would be that the user could see the origin of each assignment.

2.2 DEMPSTER'S RULE OF COMBINATION

Dempster's rule of combination requires that the categories of evidence that contribute probability mass assignments be mutually independent. The assignments contributed can be combined in any combination of pairs, triples, etc., and in any order; e.g., $(m1 \oplus m2) \oplus m3 = m1 \oplus (m2 \oplus m3) = m1 \oplus m2 \oplus m3 = m3 \oplus m2 \oplus m1$, where $\oplus$ represents combination by Dempster's rule.

Equations for Dempster's rule of combination are given in reference 2 both for combining two mass assignments and for simultaneously combining any number of mass assignments. Here we will give an algorithmic procedure for combining two assignments. (As described in section 5, however, much simpler methods of combining are used for mass assignments having special features.) We assume that the two probability mass assignments, m1 and m2, contribute mass to propositions as shown in figure 2-1. Each Cg, g = 1,....,u, and each Dh, h = 1,...,v, consist either of an original proposition Ai or of any disjunction of the n original propositions. The combining method involves taking, for each rectangle, the conjunction of the respective propositions Cg and Dh, and

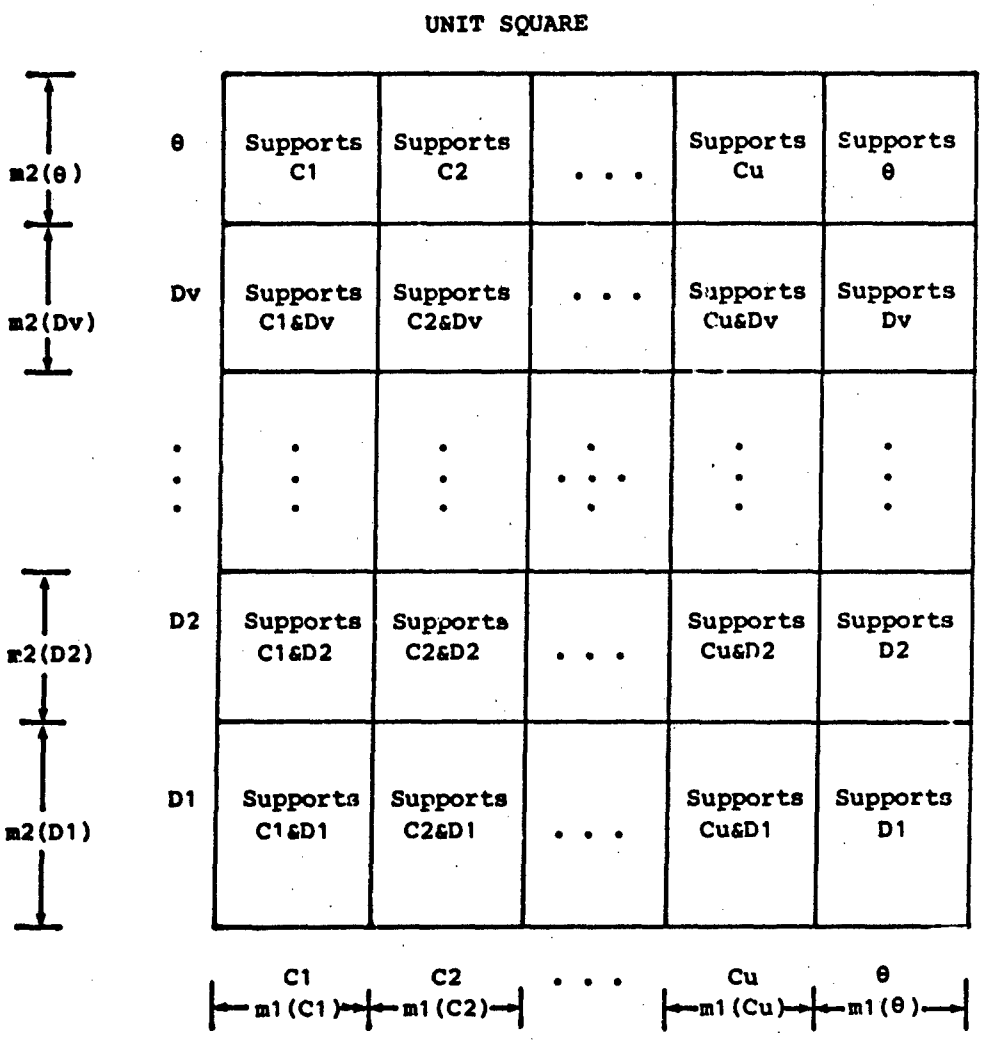


Figure 2-1. Graphical representation for combining two mass distributions.

employing the mass m1(Cg) * m2(Dh) (which is equal to the area of the rectangle) in a way determined by the value of the conjunction Cg&Dh. The procedure follows.

Step 1. Let uncommitted = 0
and let k = 0.

Step 2. For Cg = C1, ..., Cu, θ, and for Dh = D1, ..., Dv, θ,
if Cg & Dh = ~θ
(i.e., if Cg and Dh have no original proposition Ai in common),
add m1(Cg)*m2(Dh) to uncommitted;

otherwise,

if there is no Bk such that Bk = Cg&Dh; i.e., if no earlier iteration has produced a conjunction equal to Cg&Dh,

let k = k + 1
and let Bk = Cg&Dh
and let product-sum(Bk) = m1(Cg) * m2(Dh),

otherwise,

add m1(Cg) * m2(Dh) to product-sum(Bk)
where Bk = Cg&Dh.

(Note that Cg&θ = Cg and θ&Dh = Dh.)

Step 3. For each Bk resulting from step 2, let

m(Bk) = product-sum(Bk)/(1 - uncommitted).

One of the masses generated by this procedure is

m(θ) = m1(θ) * m2(θ)/(1 - uncommitted).

Because we have normalized the results with the committed mass, summing the masses over all Ek will produce m(Bk) = 1.

Note that the number of rectangles processed in the general procedure is (u + 1) * (v + 1). If an assignment is one that commits mass only to the negation of a proposition (i.e., to ~Ai for some i) and to θ, or is one that commits mass only to the original propositions and to θ, the steps are

simpler. When both assignments are of the latter type, only 3n + 1 rectangles are processed. The procedures for all cases are described in section 5 and listed in appendix A.

3. TACTICAL APPLICATIONS

In the previous year's work (ref 2), Dempster-Shafer methods were implemented in the rule-based systems ROSIE (ref 5) and STAMMER2 (ref 8) for the problem of determining ship type, but the kinds of probability mass assignments permitted were limited. An application outlined in reference 1 but not implemented was that of contact association. Since then, versions of both of these applications have been implemented in ROSIE for all valid forms of probability mass distributions.

A number of other tactical problems can be formulated in terms of mutually exclusive and exhaustive propositions, and a variety of these is listed in section 4 of reference 2. They include problems in the general categories: "What is it?," "Which one did it?," "Which one is it?," "Which partitioning is it?," "Where is it?," and "Is it or isn't it?"

3.1 PLATFORM TYPING

Our implementation of the platform typing problem is limited to surface and subsurface platforms, and ship types of lesser important are lumped together to keep the number of propositions relatively small. Each proposition Ai is of the form "Platform x is type i," and the types are listed below. We assume that each report observation of platform x is indeed an observation of that platform, as determined by the radar system or multisensor correlation system providing the input.

Ai:	Type i
A1:	Carrier
A2:	Cruiser
A3:	Destroyer
A4:	Frigate
A5:	Amphibious
A6:	Submarine (surfaced or periscope/snorkel/antenna)
A7:	Small fighting ship (e.g., corvette)
A8:	Fast attack/patrol craft
A9:	Patrol craft
A10:	Intelligence collector (e.g., AGI)
A11:	Survey/research (navy-operated)
A12:	Fleet auxiliary - medium and large
A13:	Fleet auxiliary - small

A14: Small boats (navy and commercial)
A15: Merchant
A16: Fishing
A17: Other commercial and private
A18: Debris.

In a rule-based system, the probability mass assignment is contributed by the action of a rule as the result of the rule's conditions being satisfied. (Tactical inference rules are assumed to be provided by a tactical expert, but any user can modify or extend the ruleset or at least their copy of it.) Some typical rules are outlined in table 3-1. (For each assignment there, the remaining mass is assigned to θ.) Several of these are derived from related rules created for STAMMER1 (ref 9) and STAMMER2 (ref 8).

Table 3-1. Examples of probability mass assignments for ship type. Rule conditions represent the evaluation of evidence, and rule actions assign masses.

Evidence	Mass Assignment
Region--long-term averages	<.003, .013, .04, ..., .007> to {18 types}
Initial detection range = 11	<0, 0, ..., ,.064, .055> to {18 types}
Not in merchant lane	.3 to NOT merchant
Speed = 31	<.103, .103, ..., .045, 0> to {18 types}
Course change > 5	.6 to NOT merchant
Emitter = Square Tie radar	.55 to fast-attack/patrol-craft OR small-fighting-ship
No known hostiles could have reached	.4 to merchant OR fishing OR other-commer/private
Sudden pop-up on radar	.8 to submarine

Probability mass assignments based on range and speed measurements are generated by a scheme outlined in reference 1. The scheme employs probability distributions of range and speed, respectively, for each type or category, in a manner patterned after that described in reference 6 for an emitter classification problem.

Shafer's theory on refining or coarsening a "frame or reference" (ref 4) is applicable, for example, to determining ship class (a refinement of ship type) or determining whether the ship is a combatant or noncombatant (a coarsening of ship type). Coarsening the proposition set would sometimes be useful and the computations are easily performed. Implementation of ship class computations for a ship classified "certain sub" would be practical but would not really be a refinement of a proposition set, since the contact is assumed to be a submarine. In general, a refinement of the proposition set involves a number of complexities and may not be justified for ship classification since the amount of pertinent evidence available usually is small. In cases where evidence is not plentiful, an alternate approach for dealing with possibilities within a proposition is to represent the pertinent information as descriptive assertions that can be retrieved easily in response to queries. If sufficient evidence is available, one option is to use a "dependency graph" approach like that used with the emitter classification problem in reference 6.

3.2 CONTACT ASSOCIATION

The contact association problem of primary interest is that in which detections of one or more earlier sighted platforms have ceased for a while and a new unidentified contact could be one of those platforms. We assume that the detection data provided to the rule-based system have been preprocessed and correlated to the degree feasible with a multisensor correlator-tracker scheme.

In order that the propositions be exclusive, possible duplicates must be eliminated; e.g., if two of the earlier contacts could be detections of the same platform, one must be disallowed. To make the set exhaustive, we add the

proposition that one of these candidate platforms is the current contact. We label the propositions

A1: Current contact X is platform 1.

A2: Current contact X is platform 2.

•
•
•

An - 1: Current contact X is platform n - 1.

An: Current contact X is none of these platforms.

Although here we refer to an earlier contact as platform i, in our implementation we use the label given that contact; e.g., its name, if known.

Three of the categories of evidence used earliest to derive probability mass assignments are

- Completeness of surveillance coverage (provides probability masses of proposition An and θ).
- Minimum required speed from the last report of platform i to the first report of contact X. (Consistency of this speed with platform i's last known speed and/or type, when known.)
- Difference between platform i's last known course and the course from the last position of platform i to the first position of contact X. (Knowledge of platform i's type is also employed when known).
- Initial radar detection range of contact X. (Consistency with platform i's initial detection range or type, when known.)

The latter three categories contribute mass also to An as a decreasing function of the maximum mass assigned to a candidate platform.

Note that measurements of the new contact's speed and course are not included in the above evidence. If the initial detection range is not used and no ship-type rules involving other platforms are employed, the combined mass assignment will be independent of the combined assignment for ship type. Later kinds of evidence that can be used for contact association generally will be the same ones used in making decisions about the contact's type, although the method of deriving probability mass assignments from them would be different. Because of this overlap in evidence, pursuing the association

process beyond this initial stage may be unrewarding. Limiting contact association calculations to the first three categories above but continuing to update ship-type calculations with new evidence should often be sufficient and would avoid dependence between the two sets of conclusions. If desired, the result of the association could be converted into an assignment concerning ship type (in the manner shown in the next section) and could be combined with the others concerning ship type. One the other hand, if the user is concerned about enemy ship movements and an attack is unlikely, the mutual evidence would be better used in contact association computations, although the process could become considerably more complex.

3.3 USE OF CONTACT ASSOCIATION RESULTS IN PLATFORM TYPING

The propositions concerning the possible association of the new contact with n-1 candidate platforms are

Pk: The new contact is platk (k = 1,...,n - 1)

Pn: The new contact is none of these.

The procedures we have programmed for converting evidence (the first three categories of the four in section 3.2) into mass assignments for contact association result in a combined mass assignment of the form

$$m'(P1),\ \ldots,\ m'(Pn),\ m'(\theta')$$

where θ' is the disjunction over the n propositions.

Let ty(platk) = type of plati if known
= unknown otherwise.

Let Ai represent the proposition: The new contact is typei. The mass assignment given by

$$m(Ai) = \sum_{ty(platk)=typei} m'(Pk)$$

and

$$m(\theta) = m'(\theta') + m'(Pn) + \sum_{ty(platk)=unknown} m'(Pk)$$

can then be combined with other mass assignments concerning the type of the new contact.

4. USER INTERFACE

The essential user interface feature is the presentation of results of calculations. The presentation for each set of propositions consists of a list of triples; e.g., for contact type a trip e would be type i, the measure s(type i) of evidence in support of, and the measure s(NOT type i) of evidence against. In our experimental system, we in addition name the most-likely proposition, based on the differences s(prop i) - s(NOT prop i). In this system also, computations (and the corresponding display of results) are made only upon request by the user. Later problem formulations might include some in which alerts require system-invoked calculations, but this could be accomplished easily with the rules that invoke the calculations.

Although the user is informed when a tactical interface rule fires, he may later wish to see the history of rule firings concerning some set of propositions. This is easily done in most rule-based systems, especially in ROSIE, by any user knowledgeable about the system. However, to make it easier for this user and especially to help the new user, a mechanism for displaying the history in a readable form is desirable. In our implementation, after the system displays the results of the requested computations, it offers to display the history concerning that set of propositions. For propositions about ship type, for example, if the user replies "y" to the offer, the names of the rules that fired are listed in the order fired and the probability mass assignments they contributed are displayed in a readable form. The user also can ask the system to describe a rule by typing "tell about rule-name."

A partial knowledge of Dempster-Shafer methods is needed by the user. At a minimum, he must understand the significance of the measurement pairs listed as a result of computations. It is desirable that he also have a basic understanding of how the probability mass assignments are combined into one and how, from this one assignment, the measurement pairs for each proposition are derived. The user who modifies or adds rules that contribute mass assignments should attempt to maintain independence among categories of evidence and must be able to represent his measures of confidence, based on the evidence, in the form of a legitimate probability mass assignment. When expanding a system

with new sets of propositions, he must see that the propositions are mutually exclusive and exhaustive.

The necessary and useful information about Dempster-Shafer methods should be available to the user via well written documentation. In addition, a "help" feature summarizing the various concepts should be built into the system, perhaps even with references to sections in the manual.

We mentioned above the need for representing confidences as legitimate probability mass assignments. As described in section 3.4 of reference 1, the user will tend to give his confidence in each proposition B in terms of the support s(B) instead of the probability mass m(B). [Recall, for example, that $s(A1 \vee A2) = m(A1) + m(A2) + m(A1 \vee A2)$.] If no two of the propositions (except θ) to which he gives weight overlap (i.e., the conjunction of no two is also a proposition), then the respective probability masses are simply equal to the respective support values; i.e., $m(B) = s(B)$ for each proposition B. If any two do overlap, a conversion must be made. For example, if, based on particular evidence, the user says he is 60% sure that proposition A3 is not true and 80% sure that proposition A5 is not true, we can express this as $s(\tilde{}A3) = .6$ and $s(\tilde{}A5) = .8$. The corresponding mass assignment is $m(\tilde{}A5) = .2$, $m(\tilde{}(A3 \vee A5)) = .6$, and $m(\theta) = .2$.

5. IMPLEMENTATION IN ROSIE

5.1 PROGRAMMING IN ROSIE

Rules for tactical interference, confidence computations, and other support functions were implemented in Version 1.3 of ROSIE.

A "rule" in its traditional form has one or more conditions, connected by ands and ors, and has one or more actions. In practice, conditions are not necessary and, in ROSIE, the actions may have rules imbedded within them. Most rules can be written for ROSIE in an Englishlike form easily read by a nonuser of ROSIE. For example, a simple, conditionless rule that enters data is: Assert Echo II's escort was sunk at 1300 by Perry with Harpoon missiles.

Rulesets can be written while in ROSIE or beforehand by using any text editor. In the latter case, the user, while at the top level of ROSIE, types: parse filename and load filename. The parse action creates a filepackage: filename.map, filename.parse, and filename.text. (A "program file" consists of these three directory files, plus the file filename.compile if the program is compiled.) Any further editing usually is more efficiently performed by using ROSIE's editor.

Single rules can be entered at the top level of ROSIE, in response to a prompt consisting of a line number in angle brackets, or they can be entered from a program file. Rulesets are always entered via program files. The different kinds of rulesets are outlined below. All accept parameters, which are usually passed to the rules in prepositional phrases. When a ruleset is invoked, the parameter name following the preposition in the ruleset header is bound to the parameter preceded by the same preposition.

- A procedure ruleset is used to perform tasks; e.g., to modify the data base or to coordinate the calling of other rulesets. An example is: Prioritize_threats to that snip for hostile contacts within 200 NM.
- A generator ruleset returns an element. It is invoked with a description matching the relational form of the ruleset header. An example is: Display the estimated_distance of patrol #4 from contact #6.

- A predicate ruleset determines the truth or falsehood of a relationship among elements. It is invoked by asking about a relationship whose relational form matches that of the predicate.

- A system ruleset is a procedure, function, or predicate ruleset that is written in the implementation language (currently Interlisp).

Single rules in a program file; i.e., those not in a ruleset, are evaluated at the time of loading. These typically are used to enter initial data.

ROSIE has a number of input/output actions for reading and writing directory files, for communicating with the user or with other ports, etc. It also has a pattern-matching capability that permits the binding of variables to substrings.

5.2 CONFIDENCE SETS

In our implementation, all of the probability mass assignments made concerning a particular set of propositions about a contact or other object are collected under a "confidence set." The computationally important kinds of mass assignments are separated into three categories under a confidence set by assertions of the form 'The [CategoryName]-assignments of [confidence set] is <assignment 1, ..., assignment j, ...>'. The categories of assignments and an example of each kind of assignment are given in table 5-1.

Table 5-1. Mass-assignment categories and examples.

[CategoryName]-Assignments	Example of Assignment j
Special-assignments	<.04, .06, .02, .015, 012, ..., .007>
Neg-prop-assignments	<prop4, .3>
General-assignments	<<prop2, prop3, prop5>, .2>, <<prop7>, .1>>

The "special assignments" are those contributing mass only to the n original propositions; i.e., not to disjunctions, and to θ. (See section 3.2 of ref 2.) The "neg-prop-assignments" are those contributing mass only to the negation of a single original proposition and to θ. In the example, mass .3

is assigned to ~prop4 and, by default, mass .7 is assigned to θ. A "general assignment" is any not fitting either of these categories.

A "type-conf-set" is created for a contact of unknown type when a mass assignment is first made as the result of the firing of a rule acting on evidence about platform type. At that time, an assignment consisting of "type-a priori-masses" is also made, based on the long-term average proportion of ships of each type in the region. The a priori assignment becomes a member of the tuple known as "the special-assignments of type-conf-set #j," where j is an integer designating that particular confidence set.

A "correl-conf-set" is created for a contact when a ruleset written for that purpose determines that there is at least one earlier sighted platform that could be but is not certain to be that platform. To simplify the experiments, we will assume that the type of each is known, but in general, while it is desirable that its description be fuller than the contact's, it is not necessary. The "prop-set" of a correl-conf-set consists of the platforms that were asserted to be "reachables" of the contacti based on position and physical attributes. (If two could be the same platform, one must be excluded in order that the propositions be exclusive.) The prop-set also includes as the final proposition "none-of-above." A platform is removed from the proposition set if it is later found to be an impossible association; e.g., if it is later sighted elsewhere.

5.3 MASS ASSIGNMENT BY RULE ACTION

Rule actions that convert mass assignments into the form of one of the three categories of assignments use the procedure ruleset "To assign-type-confidences of masses-or-data to type-choices for track" or "To assign-correl-confidences of masses to plat-choices for track." These perform the chores they have in common by using the procedure ruleset: "To assign-confidences of masses to prop-choices for confidence-set." The rulesets (in the file package, "K-CONF-ASSIGN" in appendix A) convert the data given as the masses and the prop-choices (type-choices or plat-choices) into a form more convenient for calculations. The result, a tuple, becomes a member of the tuple containing all assignments of that category for a confidence set (the

type-conf-set or the correl-conf-set of the track). The relationships between the formats of the input data (masses and prop-choices) and the resulting assignment category (i.e., the resulting [CategoryName]-assignments of the confidence-set), are shown in table 5-2. The "prop-set" is {A1, A2, ..., An}, where Ai = typei or plati, respectively, for the two applications.

A prop-choice tuple containing *OR* contributes mass to a single disjunction of original propositions and to θ, while one containing *NOR* contributes mass to the negation of a disjunction of original propositions and to θ. The negation of the disjunction is converted into the longer complement set for computations.

The formats for storing the various assignments (as attributes of confidence sets) are considerably different from those for creating the assignment by rule action. For example, the action of a rule may be "go assign-type-confidences of .2 to <*NOR*, <A2, A4>> for the current-track" and the result will be, for n = 8, that the tuple <<A1, A3, A5, A6, A7, A8>, .2> is added to the general-assignments of that track's type-conf-set. If the action of the rule is "go assign-type-confidences of <.2, .4> to <merchant, fishing> for the current-track," the tuple <0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, .2, .4, 0, 0> will be added to the "special-assignments" of its type-conf-set, assuming 18 type categories.

Storage formats were designed for convenience in computations because combining operations by using the assignments may be done several times, as new evidence is received. Each assignment could be used in computations only once, since each new assignment can be combined with the stored combination of earlier assignments; however, we choose to allow recall of an assignment. (Also, later grouping of like types before combining increases efficiency, as described in the next section.) As described in section 3.1, tagging the assignments would enable recall. In our implementation, this tag would be inserted as the last member of the tuple containing the assignment from the rule, although some minor revisions would be needed in the assignment format-conversion rulesets and the combining rulesets.

Table 5-2. Conversion of data from form in rule action to form for storage.

Masses [or Data]	Prop-Choices (propj = Ai for some i≤ n)	==>	[CategoryName]-Assignments
<mass1,...,massk> (k ≤ n)	<prop1,prop2,...,propk>	==>	special-assignments
mass	<*NOT*, prop>	==>	neg-prop-assignments
mass	<*OR*,<prop1,...,propk>> (k < n)	==>	general-assignments
mass	<*NOR*,<prop1,...,propk>> (k < n/2)	==>	general-assignments
<mass1,...,massk>	<<B1>,<B2>,...,<Bk>> where Bj is Ai or is any disjunction of Ai's	==>	general-assignments
For type assignments only: <range, r> or <speed, s>	<all>	==>	special-assignments

5.4 IMPLEMENTING DEMPSTER'S RULE

The strategy of combining mass assignments is implemented by a ruleset named "To generate combined-assignment of confidence-set" in the file-package named "K-CONF-MASTER." This strategy uses the following facts.

1. The combination of any number of special assignments is a special assignment. (Recall that a "special" assignment is one that assigns mass only to the n original propositions and to θ.)

2. The combination of any number of neg-prop assignments is a neg-prop assignment if the negated proposition is the same for each. (A "neg-prop" assignment is one that assigns mass to the negation of a single original proposition and to θ.)

3. Combinations involving special assignments and/or neg-prop assignments can be performed much more efficiently by rulesets tailored to the characteristics of these assignments than by rulesets capable of combining general (i.e., any) assignments.

In the two applications implemented, we are assured of having at least one special assignment. In the ship-type application, the initial assignment is based on the average ratio, in that region, of ships of each type to all ships. In the contact-correlation application, the initial assignment is based on the minimum speed computed for each earlier sighted platform (last known position) and the contact's position. The strategy used in the ruleset "To generate combined-assignment of confidence-set" therefore assumes the presence of at least one special assignment. Alternatively, we could add several additional rules to the ruleset, allowing for the absence of special assignments. This is advisable later, but meanwhile the absence of a special assignment consisting of <0, 0, ..., 0> is asserted. This is equivalent to assigning unity mass to θ and does not affect the combined result.

We have chosen to perform the combining operations in a pairwise fashion. Dempster's rule is associative and commutative, and combining can be done by groups or pairs. By using pair combining operations recursively, we simplify and minimize the rulesets that implement the combining, but we pay somewhat in efficiency.

The key rulesets performing the combining operations are listed in tables 5-3 and 5-4. Several supporting rulesets shared by these rulesets are contained in the filepackage "K-CONF-SUPPORT." The terms "special," "neg-prop," and "general" refer to the categories listed in table 5-1 and described in section 5.2.

The "special-combination" ruleset makes recursive use of the "spec-spec-combination ruleset," and the "general-combination" ruleset makes recursive use of the "gen-gen-combination" ruleset. The "like-negs-combination" ruleset is used by another ruleset in K-CONF-NEGS, named "To generate likes-combined-versions of neg-prop-assignments." The latter reduces a tuple of neg-prop assignments to one containing only one assignment per proposition represented.

Table 5-3. Pairwise-combining rulesets.

Ruleset name: "To generate [pairwise]-combination of * with *:"

Filename	[Pairwise]	*	*
K-CONF-SPECIAL-NEG	spec-spec	[special1]	[special2]
K-CONF-SPECIAL-NEG	neg-special	[neg-prop]	[special]
K-CONF-NEGS	unlike-negs	[neg-prop1]	[neg-prop2]
K-CONF-NEG-GEN	neg-gen	[neg-prop]	[general]
K-CONF-SPECIAL-GEN	special-gen	[special]	[general]
K-CONF-GENERAL	gen-gen	[general1]	[general2]

Table 5-4. Group-combining rulesets.

Ruleset name: "To generate [group]-combination of *:"

Filename	[Group]	*
K-CONF-SPECIAL-NEG	special	[1 or more specials, listed in tuple]
K-CONF-GENERAL	general	[1 or more generals, listed in tuple]
K-CONF-NEGS	like-negs	[1 or more neg-props, same prop, in tuple]

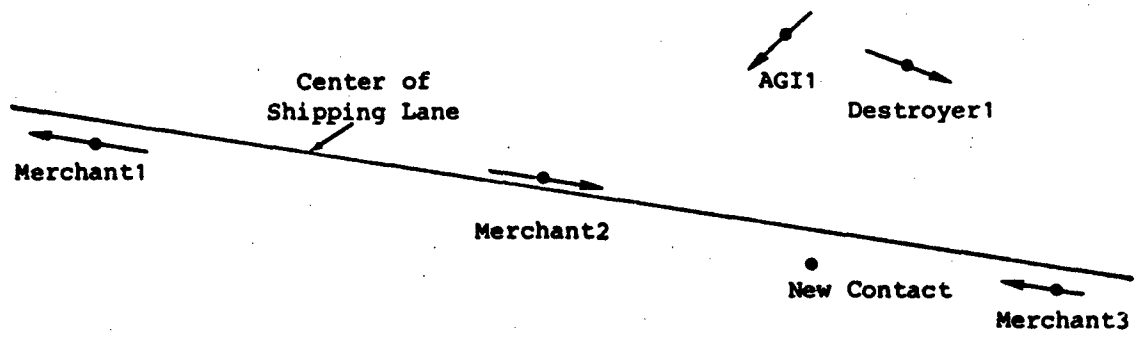
The outputs of the pairwise- and group-combination rulesets listed in tables 5-3 and 5-4 are mass assignments. The format of the combined mass assignment will depend on whether the resulting assignment is a special, neg-prop, or general assignment. The ruleset "To generate tuple-representation from B-props" in K-CONF-SUPPORT is used by those combining rulesets which produce general assignments. (Unusual combinations will sometimes produce a special or neg-prop distribution in the general format, but the final display of results will be the same no matter what the format.) The rulesets "To list_confidences over pro-con-pairs for prop-choices," "To generate special-pro-cons from masses," "To generate pro-cons from gen-assignment," and "To generate special-neg-pro-cons from mixed assignment" (all in K-CONF-OUTPUT) perform the conversion from the combined-assignment format to the list of measures of evidence for and against each original proposition.

6. EXPERIMENTS WITH TWO SCENARIOS

Tests were performed in a piecemeal fashion because of the limited storage space in a TOPS-20 implementation of ROSIE, but complete testing was possible since ROSIE has the capability of dumping and restoring databases. Sections B.1 and B.2 of appendix B give the typescripts for a scenario concerning ship type. The scenario differs from that used in reference 2 mainly in that two more general mass assignments occur. Sections B.3 and B.4 give the typescripts for a contact association scenario. The positions of the ships involved are shown in figure 6-1.

To run the scenarios under the constraint of limited space, a file "K-LOAD" was created that first loads the rulesets needed for processing reports in both scenarios and then, after the user chooses a scenario, loads the appropriate inferencing stages for the ship-type scenario and the contact association scenario, respectively. For each scenario, after the reports were processed, the data base was dumped as a file. ROSIE was exited and re-entered, the mass-assignment combining rulesets were loaded, and the data base restored. The computations were then made, as shown in sections B.2 and B.4 for the two scenarios.

Other tests of the combining rulesets were made with various combinations of assignment types. We would like to have integrated contact association results into ship-type determination by using the conversion formulas in section 3.3, but space limitations precluded this. Experiments with recalling mass assignments by using "tags" could also have been tried if more space were available.



Proposition Ai: The new contact is platform i.

i	Platform i
1	Merchant1
2	AGI1
3	Destroyer1
4	Merchant2
5	Merchant3
6	None of the above

Figure 6-1. Contact association scenario.

7. CONCLUSIONS

For tactical problems that can be formulated in terms of a set of mutually exclusive hypotheses, and if the evidence bearing on a problem can be separated into several distinct bodies, Shafer's representation scheme and Dempster's rule of combination are applicable. A temporary problem with a system employing Dempster's rule is that real-time calculations for real-time situations are not possible on today's computers. A major, continuing problem will be the system's interaction with new users, particularly one who needs to add rules that contribute probability mass assignments. A careful design of user interface mechanisms will minimize the problem, but a substantial amount of training will probably always be required. The emphasis in future work should be on developing a user-friendly interface.

REFERENCES

1. Dillard, R.A., The Dempster-Shafer Theory Applied to Tactical Data Fission in an Inference System, in Proceedings of the Fifth MIT/ONR Workshop on C3 Systems, 1982.
2. NOSC TD 545, Computing Probability Masses in Rule-Based Systems, by R.A. Dillard, Sep 1982.
3. Dempster, A.P., Upper and Lower Probabilities Induced by a Multivalued Mapping, Annals of Mathematical Statistics, no 38, p 325--339, 1967.
4. Shafer, G., A Mathematical Theory of Evidence, Princeton University Press, 1976.
5. Fain, J., et al., The ROSIE Language Reference Manual, Rand Corp. report N-1647-ARPA, Dec 1981.
6. Garvey, T.D., J.D. Lowrance, and M.A. Fishler, An Inference Technique for Integrating Knowledge from Disparate Sources, in Proceedings of the IJCAI 7, v 1, p 319--325, Aug 1981.
7. Barnett, J.A., Computational Methods for a Mathematical Theory of Evidence, in Proceeding of the IJCAI 7, v 2, p 868--875, Aug 1981.
8. NOSC TD 298, STAMMER 2: A Production System for Tactical Situation Assessment, v 1 and 2, by D.C. McCall et al., Oct 1979.
9. NOSC TD 252, STAMMER: System for Tactical Assessment of Multisource Messages, Even Radar, by R.J. Bechtel and P.H. Morris, May 1979.

APPENDIX A

TEXT FILES OF RULESET FILEPACKAGES

CONTENTS

```
[: K-NEWREPORT  Created 11-May-83 08:49:47, edit by DILLARD :]

[rule 1] Let the report-number be 0.

To new-report:

[1] Let the report-number be the report-number + 1.

[2] Load the name {"Krpt", the report-number}.

End.


To process_report with report:

[1] Go print_an_outline of the report.

[2] Send {return, " ... running-rules on report ...", return, return}.

[3] Go run_TSA-Rules on the report.

[4] Send {return, return,
"To receive the next report, type:  new-report.",
return, return}.

End.


To generate track of plat:

[1] If there is a track (tr)
such that the plat is a platform of tr,
produce tr,
otherwise,
create a track (tr)
and assert tr is a track of the plat
and let the platform of tr be the plat
and produce tr.

End.


To print_an_outline of rpt:

Private trk, plat.

[1] If there is a track (t)
such that the rpt is a report in t,
(let the trk be t
and if there is a platform (p) of t,
let the plat be p),
otherwise,
send {return, "No tracks reported in ", the rpt, return}
and return.
```

```
[2] Send {return, the rpt, return}.
[3] Send {"   Platform:  ", the plat, return}.
[4] Send {"   Track:     ", the trk, return}.
[5] Send {return}.
End.
```

```
[: K-TSA-RULES  Created 19-May-83 14:11:36, edit by DILLARD :]

[rule 1] Let continuous be the status of surface-search-radar.

To Run_TSA-rules on current-report:

Private current-track.

[1] If there is a track (t)
such that the current-report is a report in t,
let the current-track be t,
otherwise,
send {return, "Please assert:  ", the current-report}
and send {
"        is a report in (the track of [contact-name-or-label].
Then type:  Go run_TSA-rules on ", the current-report, ".", return}
and return.

[2] If there is a type-conf-set (tcs) of the current-track
and the platform_type of the current-track ~= unknown,
send {return,
"Speculations about the type of this platform are being erased,
because the type is now known.", return}
and forget about tcs
and for each track (tr) such that
   the current-track is a reachable of tr,
forget about (the correl-conf-set of tr).

[3] For each track (tr) such that
   the current-track is a reachable of tr,
if there is a platform (p) of the current-track
and (the abs_value of
     (the min_speed of <the current-report, the last-report of tr>))
       > (the max-speed of p),
deny the current-track is a reachable of tr
and the current-track is a likely-continuation of tr
and forget about (the correl-conf-set of tr)

and (if p is a hostile-reachable of tr,
deny p is a hostile-reachable of tr
and send {return, "A minor rule fires:  "}
and send {return,
"Because of the new report of ", p,
", the assertion that ",
p, " is a hostile-reachable of ", tr,
" is now denied.",
return, return}

and (if there is no hostile-reachable of tr,
assert Belated-Not-Known-Hostile is a fired-rule of tr
and send {return, "Belated-Not-Known-Hostile Rule fires:  "}
```

```
and send {return,
"The position of the first-report of ", tr,
" is not within ", "reach of any platform identified as hostile.",
return, return}
and go assign-type-confidences of .4 to
   <*OR*, <merchant, fishing, other-commer/private>> for tr)

and for each inactive-track (iat) of p,
deny (iat) is a reachable of tr
    and (tr) is a likely-continuation of (iat)).

[4] Go run_contact-assoc_rules on (the current-report)
   of the current-track.

[5] If the platform_type of the current-track ~= unknown,
return.

[6] If the first-report of the current-track = the current-report
and there is no hostile-reachable of the current-track,
assert Not-Known-Hostile is a fired-rule of the current-track
and send {return, "Not-Known-Hostile Rule fires:  "}
and send {return,
"The position of the first-report of ", the current-track,
" is not ", "within reach of any platform identified as hostile.
"}
and go assign-type-confidences of .4 to
   <*OR*, <merchant, fishing, other-commer/private>>
   for the current-track.

[7] If (the platform of the current-track) is a unique-name
and there is a correl-conf-set (ccs) of the current-track,
forget about ccs.

[8] If the first-report of the current-track = the current-report
and there is a sensor (s) of the current-report
and s = radar,
assert the current-report is a radar-popup.

[9] If the current-report is a radar-popup
and there is a signal-strength (ss) of the current-report
and ss = strong
and the range of the current-report < 8
and continuous is a status of surface-search-radar,
let surface be the mode-if-sub of the current-report
and assert Sub-Surfacing is a fired-rule of the current-track
and send {return, "Sub-Surfacing Rule fires:  "}
and send {return,
"The mode-if-sub of ", the current-report, " is surface."}
and go assign-type-confidences of <.8> to <submarine>
   for the current-track.
```

```
[10] If the current-report is a radar-popup
and continuous is a status of surface-search-radar
and (there is no signal-strength of the current-report
or the  signal-strength of the current-report ~= strong)
and there is a range (r) of the current-report,
assert Radar-Popup-Range is a fired-rule of the current-track
and send {return,
"Radar-Popup-Range Rule fires for ", the current-track,
".",  "  (Range = ", r, ")", return}
and let r be the initial-range of the current-track
and go assign-type-confidences of <range, r> to <types>
   for the current-track.

[11] If there is a speed (s) of the current-report
and every fired-rule of the current-track ~= speed,
assert Speed is a fired-rule of the current-track
and send {return,
"Speed Rule fires for ", the current-track, ".",
"   (Speed = ", s, ")", return}
and let s be the initial-speed of the current-track
and go assign-type-confidences of <speed, s> to <types>
   for the current-track.

[12] If there is a course (c1) of the current-report,
for each  report (rpt2) in the current-track,
if rpt2 ~= the current-report
and every fired-rule of the current-track ~= course-changed
and  there is a course (c2) of rpt2
and the Course_Difference of <c1, c2> > 5,
assert Course-Changed is a fired-rule of the current-track
and send {return,
"Course-Changed Rule fires for ", the current-track, ":"}
and send {return, " ", the current-report, " course:  ",
the course of the current-report, return,
" ", rpt2, " course:  ", the course of rpt2,
return}
and go assign-type-confidences of .6 to <*NOT*, merchant>
   for the current-track.

[13] If there is an emitter (e) of the current-report
and e = square-tie-radar
and every fired-rule of the current-track ~= square-tie-radar,
assert square-tie-radar is a fired-rule of the current-track
and send {return,
"Square-Tie-Radar Rule fires for ", the current-track, ".",
return}
and go assign-type-confidences of .55 to
   <*OR*, <fast-attack/patrol-craft, small-fighting-ship>>
      for the current-track.
```

```
[14] If there is a latitude (lat) of the current-report
and there is a longitude (lon) of the current-report
and the range-to-lane-center of
      <lat, lon> to <32, 170, 30, 180> < 10,
assert Merchant-Lanel is an occupied-lane of the current-report
and send {return, "A minor rule fires:  "}
 and send {return,
"The position of ", the current-report, " is inside a Merchant-Lane.
", return}.

[15] If there is no occupied-lane of the current-report
and every fired-rule of the current-track ~= outside-all-lanes,
assert Outside-All-Lanes is a fired-rule of the current-track
and send {return, "Outside-All-Lanes Rule fires:  "}
and send {return,
"The position of ", the current-report, " of ", the current-track,
" is outside all merchant lanes.",
return}
and go assign-type-confidences of .3 to <*NOT*, merchant>
   for the current-track.

End.
```

```
[: K-REACHABLES  Created 17-May-83 15:22:27, edit by DILLARD :]

To run_Contact-Assoc_Rules on current-report of current-track:

Private two-pos-spd, two-pos-crs.

[1] If the first-report of the current-track = the current-report
and (the platform of the current-track) is
    not_fully_identified
and there is a time (ct) of the current-report,
for each track (trk),
if trk ~= the current-track
and the current-track is not_platform-unlike trk
and there is a last-report (rpt) of trk
and there is a time (xt) of rpt
and xt < ct,
let the two-pos-spd be (the distance of <the current-report, rpt>)
      /(the time_difference of <ct, xt>)
and if the two-pos-spd < the max-speed of (the platform of trk),
assert trk is a reachable of the current-track
and let the two-pos-crs be
    the average-course of <rpt, the current-report>
and the reachables of the current-track be
    the concatenation of <<trk, the two-pos-spd, the two-pos-crs>>
        with (the reachables of the current-track).

[2] If the first-report of the current-track = the current-report
and there is a platform (plat) of the current-track
and plat is unique-name
and there is no inactive-track of plat
and there is a time (ct) of the current-report,
for each track (trk),
if trk ~= the current-track
and the current-track is not_platform-unlike trk
and there is a last-report (rpt) of trk
and there is a time (xt) of rpt
and xt < ct,
let the two-pos-spd be (the distance of <the current-report, rpt>)
      /(the time_difference of <ct, xt>)
and if the two-pos-spd < the max-speed of the plat,
assert trk is a reachable of the current-track.

[3] If the first-report of the current-track = the current-report
and there is a reachable of the current-track,
send {return, "A minor rule fires:  "}
and send {return,
"The position of the first-report of ", the current-track,
" can be ", "reached from the position of the last-report of:",
return}
and for each reachable (trk) of the current-track,
send {"     ", trk, return},
and send {return}.
```

```
[4] If the first-report of the current-track = the current-report
and there is a platform (p) of the current-track
and there is an inactive-track (iat) of p
and there is a time (ct) of the current-report,
for each track (trk),
if (iat) is a reachable of trk
and there is a last-report (rpt) of trk
and there is a time (xt) of rpt
and xt < ct
and ((the distance of <the current-report, rpt>)
      /(the time_difference of <ct, xt>))
     < the max-speed of p,
 assert trk is a reachable of the current-track
 and send {return, "A minor rule fires:  "}
 and send {return,
"The position of the first-report of ", the current-track,
" can be ", "reached from the position of the last-report of
", trk, ".", return,
"The platform ", p, " of ", the current-track, " could earlier
have reached ", trk, " from ", iat, ".",
return, return}.

[5] If the first-report of the current-track = the current-report,
for each reachable (trk) of the current-track,
if the platform_type of the current-track = unknown
and there is a platform (p) of trk
and hostile is an ID of p,
 assert p is a hostile-reachable of the current-track
 and send {return, "A minor rule fires:  "}
 and send {return,
"A platform that could have reached ", the current-track,
" is hostile:
   ", p, "
", return}.

[6] If the first-report of the current-track = the current-report,
for each reachable (tr) of the current-track,
(if the current-track is initially-course-consistent with tr,
assert the current-track is initially-course-consistent with tr
and send {return,
the current-track, " is consistent in course with ", tr, ".",
return, return})
and (if the current-track is initially-speed-consistent with tr,
assert the current-track is initially-speed-consistent with tr
and send {return,
the current-track, " is consistent in speed with ", tr, ".",
return, return}).

End.
```

```
To use_speed&course to later-track:

Private propset, speeds, courses, plat, speed, course,
spd-soundness-measure, crs-soundness-measure, spd-measures,
crs-measures.

[1] If there is a correl-conf-set (cs) of the later-track,
let the propset be the prop-set of cs
and the speeds be the minimum-speeds of cs
and the courses be the average-courses of cs.

[2] Let the spd-measures be <>.

[3] Let the crs-measures be <>.

[4] For each integer (i) from 1 to (the length of the propset) - 1,
let the plat be the member at i of the propset
and the speed be the member at i of the speeds
and the course be the member at i of the courses
and the spd-soundness-measure be
    the speed-soundness of (the speed) for the plat
and the spd-measures be the concatenation of
   (the spd-measures) with <the spd-soundness-measure>
and the crs-soundness-measure be
    the course-soundness of (the course) for the plat
and the crs-measures be the concatenation of
   (the crs-measures) with <the crs-soundness-measure>.

[5] Go assign-correl-confidences of
   (the normalized-soundness-measures from the spd-measures)
      to (the propset) for the later-track.

[6] Go assign-correl-confidences of
   (the normalized-soundness-measures from the crs-measures)
      to (the propset) for the later-track.

End.


To compare_plat-sizes with later-track:

Private propset, size-measures, init-range, plat,
size-soundness-measure.

[1] If there is a correl-conf-set (cs) of the later-track,
let the propset be the prop-set of cs.

[2] Let the size-measures be <>.

[3] Let the init-range be the initial-range of the later-track.
```

```
[4] For each integer (i) from 1 to (the length of the propset) - 1,
let the plat be the member at i of the propset
and the size-soundness-measure be
    the size-soundness of (the init-range) for the plat
and the size-measures be the concatenation of
   (the size-measures) with <the size-soundness-measure>.

[5] Go assign-correl-confidences of
   (the normalized-soundness-measures from the size-measures)
      to (the propset) for the later-track.

End.


To generate course-soundness of two-posit-crs for platform:

Private course-diff, sigma.

[1] If there is a course (crs) of
  (the last-report of (the track of the platform)),
let the course-diff be the course_difference of
    <the two-posit-crs, crs>,
otherwise,
produce 3.

[2] If there is a type (ty) of the platform
and ty = merchant,
(if the course-diff > 12,
produce 0,
otherwise,
let the sigma be 3),
otherwise,
if the course-diff < 8,
let the sigma be 5.2,
otherwise,
produce 3.

[3] Produce 25 * (the normal-density of (the course-diff)
    with 0 for the sigma).

End.


To generate speed-soundness of two-posit-spd for platform:

Private ratio.
```

```
[1] If there is a speed (sp) of
   (the last-report of (the track of the platform)),
let the ratio be the two-posit-spd / sp
and if there is a type (ty) of the platform
and ty = merchant,
(if the ratio < .7,
 produce 0)
and (if the ratio < .8,
 produce 2.5)
and (if the ratio < .9,
 produce 5)
and (if the ratio < .95,
 produce 7.5)
and (if the ratio < 1.05,
 produce 10)
and (if the ratio < 1.1,
 produce 7.5)
and (if the ratio < 1.2,
 produce 5)
and (if the ratio < 1.3,
 produce 2.5)
and produce 0,
otherwise,
(if the ratio < .5,
 produce 2.5)
and (if the ratio < .7,
 produce 5)
and (if the ratio < .9,
 produce 7.5)
and (if the ratio < 1.1,
produce 10)
and (if the ratio < 1.2,
 produce 7.5)
and (if the ratio < 1.3,
 produce 5)
and (if the ratio < 1.4,
 produce 2.5)
and produce 0.

[2] If there is a type (ty) of the platform
and ty = merchant,
produce 20 * (the normal-density of (the two-posit-spd)
    with 21 for 6).

[3] Produce 2.5.

End.
```

```
To generate size-soundness of init-range for platform:

[1] If there is a type (ty) of the platform
and ty = submarine,
(if the init-range < 6,
produce 7.5)
and (if the init-range > 22,
produce 2.5)
and produce 5.

[2] If there is an initial-range (rangel) of (the track of the platform),
produce 25 * (the normal-density of (rangel - the init-range)
     with 0 for 3),
otherwise,
if there is a type (ty) of the platform,
produce the type-size-soundness of ty with the init-range,
otherwise,
produce 5.

End.


To generate type-size-soundness of type with range:

Private mean, sigma.

[1] If there is a mean-det-range (r) of the type,
let the mean be r
and the sigma be 5,
otherwise,
if the type = other-commer/private,
let the mean be 20
and the sigma be 10,
otherwise,
if the type = small-boat,
let the mean be 6
and the sigma be 3,
otherwise,
let the mean be 20
and the sigma be 5.

[2] Produce 20 * (the normal-density of (the range) with (the mean)
   for the sigma).

End.
```

```
To generate normalized-soundness-measures from measures:

Private output, sum, mlength, max, none-weight, norm-factor.

[1] Let the output be <>.

[2] Let the sum be the tuple-sum of the measures.

[3] Let the mlength be the length of the measures.

[4] Let the max be 0.

[5] For each member (i) of the measures,
if i > the max,
let the max be i.

[6] Let the none-weight be .7 - .07 * the max.

[7] Let the norm-factor be (.7 - the none-weight) / the sum.

[8] For each integer (i) from 1 to the mlength,
let the output be the concatenation of (the output) with
   <the norm-factor * (the member at i of the measures)>.

[9] Let the output be the concatenation of (the output) with
   <the none-weight>.

[10] Produce the output.

End.
```

```
[: K-SUPPORT  Created 23-May-83 08:04:06, edit by DILLARD :]

To generate platform_type of track:

[1] If there is a platform (p) of the track
and there is a type (ty) of p
produce ty.

[2] Produce unknown.

End.


To generate reachables of track:

[1] Produce <>.

End.


To decide ship1 is described_unlike ship2:

[1] If the ship1 is a unique-name
and the ship2 is a unique-name
and the ship1 ~= the ship2
conclude true.

[2] If there is an ID (ID1) of the ship1
and there is an ID (ID2) of the ship2
and ID1 ~= ID2,
conclude true.

[3] If there is a type (t1) of the ship1
and there is a type (t2) of the ship2
and t1 ~= t2,
conclude true.

[4] If there is a class (c1) of the ship1
and there is a class (c2) of the ship2
and c1 ~= c2,
conclude true.

[5] Conclude false.

End.
```

```
To decide track1 is not_platform-unlike track2:

[1] If there is a platform (plat1) of the track1
and there is a platform (plat2) of the track2,
and plat1 is described_unlike plat2,
conclude false.

[2] Conclude true.

End.


To generate description_similarity of ship1 to ship2:

[1] If the ship1 = the ship2, produce 15.

[2] If there is a class (c1) of the ship1
and there is a class (c2) of the ship2
and c1 = c2
produce 10.

[3] If there is a type (t1) of the ship1
and there is a type (t2) of the ship2
and t1 = t2
produce 5.

[4] Produce 0.

End.


To decide ship is not_fully_identified:

[1] If the ship is a unique-name,
conclude false.

[2] Conclude true.

End.


To generate max-speed of platform:

Private plat-type.

[1] If there is no type of the platform,
produce 40,
otherwise let the plat-type be the type of the platform.

[2] If the plat-type = fast-attack/patrol-craft,
produce 45.
```

```
[3] If (the plat-type = carrier
 or the plat-type = cruiser
 or the plat-type = destroyer),
produce 35.

[4] Produce 30.

End.


To generate correl-conf-set of track:

[This ruleset fires only once for each track unless Rule [2] or [3]
of "To Run_TSA-rules on current-report" erases the confidence set
because of information that eliminates a proposition.]

Private conf-set, propositions, speeds, courses, trk, two-pos-spd,
two-pos-course.

[1] If there is a correl-conf-set (ccs)
such that ccs is a correl-conf-set for
    (the platform of the track),
produce ccs
and return,
otherwise,
create a correl-conf-set (ccs)
and assert ccs is a correl-conf-set of the track
and ccs is a correl-conf-set for (the platform of the track)
and let the conf-set be ccs.

[2] Let the propositions be <none-of-the-above>.

[3] Let the speeds be <>.

[4] Let the courses be <>.

[5] For each member (m) of
   (the reachables of the track),
let the trk be the member at 1 of m
and if the trk is a reachable of the track
and the trk is not_platform-unlike the track,
let the propositions be the concatenation of
   <the platform of the trk> with the propositions
and the two-pos-spd be the member at 2 of m
and the speeds be the concatenation of
   <the two-pos-spd> with the speeds
and the two-pos-crs be the member at 3 of m
and the courses be the concatenation of
   <the two-pos-crs> with the courses.
```

```
[6] Let the minimum-speeds of the conf-set be the speeds.

[7] Let the average-courses of the conf-set be the courses.

[8] Let the prop-set of the conf-set be the propositions.

[9] Go assign-correl-confidences of <.6> to <none-of-the-above>
      for the track.

[10] Produce the conf-set.

End.


To generate normal-density of arg with mean for sigma:

Private constant, dummy.

[1] Let the dummy be (the arg - the mean) / the sigma.

[2] If the dummy > 10
or the dummy < -10,
produce 0.

[3] Let the constant be 1 / (3.141593 * 2) ** .5.

[4] Produce the constant / 2.718282 ** (.5 * the dummy ** 2).

End.


[rule 1] Let the mean-det-range of carrier be 27
and the mean-det-range of cruiser be 27
and the mean-det-range of destroyer be 27
and the mean-det-range of frigate be 27
and the mean-det-range of amphibious be 27
and the mean-det-range of survey/research be 27
and the mean-det-range of fleet-auxil-med/lrg be 27
and the mean-det-range of merchant be 23
and the mean-det-range of AGI be 20
and the mean-det-range of small-fighting-ship be 17
and the mean-det-range of fleet-auxil-small be 17
and the mean-det-range of fishing be 16
and the mean-det-range of fast-attack/patrol-craft be 12
and the mean-det-range of patrol-craft be 12.
```

```
[: K-MATH  Created 23-May-83 16:05:15, edit by DILLARD :]

To generate subtended_angle of latlons:

Private alat, alon, blat, blon, k1, k2, k3.

[1] Let the alat be the member at 1 of the latlons
and the alon be the member at 2 of the latlons
and the blat be the member at 3 of the latlons
and the blon be the member at 4 of the latlons.

[2] If the alon = the blon
produce the abs_value of (the alat - the blat).

[3] If the abs_value of (the blon - the alon) < .7
produce ((((the cosine of the alat) * (the blon - the alon)) ** 2)
   + ((the blat - the alat) ** 2)) ** .5.

[4] Let the k1 be the cosine of (the alat + the blat)
and the k2 be the cosine of (the blat - the alat)
and the k3 be the cosine of (the blon - the alon)
and produce the arccosine of
(((the k3 * (the k1 + the k2)) + (the k2 - the k1)) / 2).

End.


To generate ABS_value of numberC:

[1] If the numberC < 0  produce the negation of the numberC.

[2] Produce the numberC.

End.


To generate Time_Difference of times:

Private atime, btime.

[1] Let the atime be the member at 1 of the times
and the btime be the member at 2 of the times.

[2] Match the atime against
{something (bind aday to the number),
2 nonblanks (bind ahour to the number),
2 nonblanks (bind aminute to the number)}

and match the btime against
{something (bind bday to the number),
2 nonblanks (bind bhour to the number),
2 nonblanks (bind bminute to the number)}
```

```
and if aday - bday = 0,
produce ahour + aminute / 60 - bhour - bminute / 60,
otherwise,
produce ahour + (aday - bday) * 24 + aminute / 60
   - bhour - bminute / 60.

End.


To generate Time_Delta of reports:

Private REPORTi, REPORTj.

[1] Let the REPORTi be the member at 1 of the reports
and the REPORTj be the member at 2 of the reports.

[2] Produce the time_difference of
   <the time of the REPORTi, the time of the REPORTj>.

End.


To generate Distance of reports:

Private REPORTa, REPORTb.

[1] let the REPORTa be the member at 1 of the reports
and the REPORTb be the member at 2 of the reports.

[2] If there is a latitude (alat) of the REPORTa
and there is a longitude (alon) of the REPORTa
and there is a latitude (blat) of the REPORTb
and there is a longitude (blon) of the REPORTb,
produce 60 * (the subtended_angle of
                  <alat, alon, blat, blon>).

[3] Produce 10000.

End.


To generate Min_Speed of reports:

[1] Produce (the distance of the reports)
       / (the time_delta of the reports).

End.
```

```
To generate direction of latlons:

Private bearangle, bearvar, latdif, londif.

[1] Match the latlons against
{anything, "<",
anything (bind alat to the number),
",",
anything (bind alon to the number),
",",
anything (bind blat to the number),
",",
anything (bind blon to the number),
">", anything}

and (if alat = 90 produce 180)
and (if alat = -90 produce 0)

and let the londif be blon - alon
and the latdif be blat - alat

and (if the abs_value of the londif < .7,
let the bearvar be (the cosine of alat) * the londif
and the bearangle be the arctangent of
     (the bearvar / the latdif)
and (if the bearvar < 0
   let the bearangle be 180 + the bearangle))

and (if the abs_value of the londif >= .7,
let the bearvar be
((the cosine of blat) * (the sine of the londif))
 / (the sine of
    (the subtended_angle of <alat,alon,blat,blon>))
and (if the bearvar < -1 let the bearvar be -1)
and (if the bearvar > 1 let the bearvar be 1)
and let the bearangle be the arcsine of the bearvar
and (if blat < alat
   let the bearangle be 180 - the bearangle)).

[2] If the bearangle < 0
 produce 360 + the bearangle.

[3] Produce the bearangle.

End.
```

```
To generate average-course of two-reports:

Private report1, report2.

[1] Let the report1 be the member at 1 of the two-reports.

[2] Let the report2 be the member at 2 of the two-reports.

[3] If there is a latitude (lat1) of the report1
and there is a longitude (lon1) of the report1
and there is a latitude (lat2) of the report2
and there is a longitude (lon2) of the report2,
produce the direction of <lat1, lon1, lat2, lon2>.

End.


To generate Course_Difference of Courses:

Private Q1, Q2.

[1] Let the Q1 be the member at 1 of the courses
and the Q2 be the member at 2 of the courses.

[2] If the Q2 >= the Q1,
if the Q2 > the Q1 + 180,
produce 360 + the Q1 - the Q2,
otherwise,
produce the Q2 - the Q1.

[3] If the Q1 > the Q2 + 180,
produce 360 + the Q2 - the Q1,
otherwise,
produce the Q1 - the Q2.

End.


To generate range-to-lane-center of position to lane:

Private cat, sat, can, san, cbt, sbt, cbn, sbn, cct, sct, ccn,
   scn, sinsub, arcc.

[1] Match the lane against
{anything, "<",
anything (bind alat to the number),
",",
anything (bind alon to the number),
",",
anything (bind blat to the number),
",",
anything (bind blon to the number),
">", anything}
```

```
and match the position against
{anything, "<",
anything (bind clat to the number),
",",
anything (bind clon to the number),
">", anything}

and let the cat be the cosine of alat
and the sat be the sine of alat
and the can be the cosine of alon
and the san be the sine of alon
and the cbt be the cosine of blat
and the sbt be the sine of blat
and the cbn be the cosine of blon
and the sbn be the sine of blon
and the cct be the cosine of clat
and the sct be the sine of clat
and the ccn be the cosine of clon
and the scn be the sine of clon

and let the sinsub be the sine of (the subtended_angle of
<alat,alon,blat,blon>).

[2] Let the arcc be the arccosine of ((

the cat * the can * (the cbt * the sbn * the sct - the
sbt * the cct * the scn)

- the cat * the san * (the cbt * the cbn * the sct - the sbt
* the cct * the ccn)

+ the sat * (the cbt * the cbn * the cct * the scn - the cbt
* the sbn * the cct * the ccn)

) / the sinsub).

[3] Produce 60 * the abs_value of (90 - the arcc).

End.


To generate report_count of track:

Private count.

[1] Let the count be 0.

[2] For each report in the track,
let the count be the count + 1.

[3] produce the count.

End.
```

```
To generate first-report of track:

private mintime, first-one.

[1] Let the mintime be 312359.

[2] For each report (r) in the track,
if the time of r < the mintime,
let the mintime be the time of r
and let the first-one be r.

[3] Produce the first-one.

End.


To generate last-report of track:

private maxtime, last-one.

[1] Let the maxtime be 0.

[2] For each report (r) in the track,
if the time of r > the maxtime,
let the maxtime be the time of r
and let the last-one be r.

[3] Produce the last-one.

End.
```

```
[: K-EXPLAIN  Created  7-Feb-83 07:58:28, edit by DILLARD :]

To Tell about arg:

[1] If the arg = radar-popup-range
send {return,
"The initial radar detection range is indicative of the
platform's size and contributes confidence weights over all
platform types.",
return, return}.

[2] If the arg = course-changed
send {return,
"If the course has changed significantly, then the ship
probably isn't a merchant (.6).",
return, return}.

[3] If the arg = speed
send {return,
"The speed measurement contributes confidence weights over all
platform types.",
return, return}.

[4] If the arg = outside-all-lanes
send {return,
"If the sighting is outside of all merchant lanes, then the
ship might not be a merchant. (.3).",
return, return}.

[5] If the arg = square-tie-radar
send {return,
"If the intercepted signal is a square tie radar, the contact is
likely to be a fast-attack/patrol-craft or a small-fighting-ship
(.55).",
return, return}.

[6] If the arg = not-known-hostile
send {return,
"If no earlier sighted hostile could have reached the position
of the new contact, then the contact is probably merchant, fishing,
or other-commer/private (.4).",
return, return}.

End.
```

```
[: K-CONF-ASSIGN  Created  8-Aug-83 09:34:19, edit by DILLARD :]

[Rulesets convert data from rule-action format to storage format
and store them as attributes of confidence sets.]

[rule 1] Let the type-categories be
  <carrier, cruiser, destroyer, frigate, amphibious, submarine,
   small-fighting-ship, fast-attack/patrol-craft, patrol-craft,
   intell-collector, survey/research, fleet-auxil-med/lrg,
   fleet-auxil-small, small-boat, merchant, fishing,
   other-commer/private, debris>.

[rule 2] Let the type-apriori-masses be <.003, .013, .04, .03, .04,
.02, .04, .06, .02, .015, .012, .09, .03, .01, .13, .07, .02, .007>.

To assign-type-confidences of masses-or-data to type-choices
    for track:

[1] If the type-choices = <types>,
let the special-assignments in (the type-conf-set of the track)
  be the concatenation of
   (the special-assignments in (the type-conf-set of the track))
    with <the mass-vector_derived from the masses-or-data>
and return.

[2] Let the prop-set be the type-categories
and go assign-confidences of (the masses-or-data)
       to (the type-choices)
    for (the type-conf-set of the track).

End.


To assign-correl-confidences of masses to plat-choices for track:

[1] If there is a correl-conf-set (conf-set) of the track,
let the prop-set be (the prop-set of conf-set)
and go assign-confidences of (the masses) to (the plat-choices)
   for conf-set.

End.


To assign-confidences of masses to prop-choices for confidence-set:

Private key.

[1] Let the key be the member at 1 of the prop-choices.
```

```
[2] If the key = *NOT*,
  let the neg-prop-assignments in the confidence-set
     be the concatenation of
         (the neg-prop-assignments in the confidence-set)
         with <<the member at 2 of the prop-choices,
                              the masses>>
and return.

[3] If the key = *OR*,
  let the general-assignments in the confidence-set
     be the concatenation of
         (the general-assignments in the confidence-set) with
          <<<the member at 2 of the prop-choices, the masses>>>
and return.

[4] If the key = *NOR*,
  let the general-assignments in the confidence-set
     be the concatenation of
         (the general-assignments in the confidence-set)
         with <<<the logical-negation of
                (the member at 2 of the prop-choices), the masses>>>
and return.

[5] If the key is a tuple,
  let the general-assignments in the confidence-set
     be the concatenation of
         (the general-assignments in the confidence-set)
         with <the general-conversion from (the masses)
                for the prop-choices>
and return.

[6] Let the special-assignments in the confidence-set
      be the concatenation of
          (the special-assignments in the confidence-set)
         with <the special-conversion from (the masses)
                    for the prop-choices>.

End.


To generate type-conf-set of track:

[1] If there is a type-conf-set (cs)
such that cs is a type-conf-set for (the platform of the track),
produce cs
and return.

[2] If the platform_type of the track = unknown,
create a type-conf-set (cs)
and assert cs is a type-conf-set of the track
and cs is a type-conf-set for (the platform of the track)
and let the special-assignments in cs be <the type-apriori-masses>
and produce cs.

End.
```

```
To generate mass-vector_derived from sensor-measurement:

[Produces representative mass assignments (in the special format)
for range and speed measurements.  In practice, the range of
sensor measurements should be covered.]

Private measured, measurement.

[1] Let the measured be the member at 1 of the sensor-measurement.

[2] Let the measurement be the member at 2 of the sensor-measurement.

[3] If the measured = range,
    (if the measurement <= 12
    and the measurement >= 10,
produce <0, 0, 0, 0, 0, .037, .045, .136, .136, .027, 0, 0,
   .045, .055, .023, .077, .064, .055>)
and (if the measurement <= 22
    and the measurement >= 20,
produce <.04, .04, .04, .04, .04, .059, .054, .01, .01, .071,
   .04, .04, .054, 0, .066, .049, .047, 0>).

[4] If the measured = speed,
  if the measurement < 33
 and the measurement >= 29,
produce <.103, .103, .103, .029, .008, .009, .029, .074,
   .009, .008, .008, .008, .008, .011, .034, .011, .045, 0>.

[5] Produce <0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>.

End.


To generate special-assignments in conf-set:

[1] Produce <>.

End.


To generate neg-prop-assignments in conf-set:

[1] Produce <>.

End.


To generate general-assignments in conf-set:

[1] Produce <>.

End.
```

```
To generate special-conversion from masses for prop-choices:

Private vector, choice-length.

[1] Let the vector be <>.

[2] Let the choice-length be the length of the prop-choices.

[3] For each integer (i) from 1 to the choice-length,
assert (the member at i of the masses) is a
    mass of (the member at i of the prop-choices).

[4] For each member (m) of the prop-set,
if there is a member (choice) (of the prop-choices)
   such that m = choice,
let the vector be the concatenation of (the vector)
   with <the mass of choice>,
otherwise,
let the vector be the concatenation of (the vector) with <0>.

[5] For each member (m) of the prop-choices,
deny (the mass of m) is a mass of m.

[6] Produce the vector.

End.


To generate general-conversion from masses for prop-choices:

Private new-form, count.

[1] Let the count be 0.

[2] Let the new-form be <>.

[3] For each member (gen-prop) of the prop-choices,
let the count be the count + 1
and let the new-form be
   the concatenation of (the new-form) with
     <<gen-prop, the member at (the count) of the masses>>.

[4] Produce the new-form.

End.
```

```
[: K-CONF-SUPPORT  Created  1-Feb-83 13:12:56, edit by DILLARD :]

To generate logical-negation of neg-prop:

[Gives a tuple containing only complement members of the
 PROP-SET.  Variable neg-prop can be a tuple or an
 element.]

Private longprop.

[1] Let the longprop be <>.

[2] If the neg-prop is a tuple,
(for each member (orig-prop) of the prop-set,
if there is no member (m) of (the neg-prop)
      such that m = orig-prop,
let the longprop be the concatenation
   of (the longprop) with <orig-prop>)
otherwise,
(for each member (orig-prop) of the prop-set,
if the neg-prop ~= orig-prop,
let the longprop be the concatenation
   of (the longprop) with <orig-prop>).

[3] Produce the longprop.

End.


To generate tuple-sum of tuplex:

[We can't sum over "each member (m) of the tuplex" because that
requires unique members.]

Private sum.

[1] Let the sum be 0.

[2] For each integer (i) from 1 to (the length of the tuplex),
let the sum be the sum +
   the member at i of the tuplex.

[3] Produce the sum.

End.


To generate theta-mass of gen-assignment:

Private interim.

[1] Let the interim be 1.
```

```
[2] For each member (xtuple) of the gen-assignment,
let the interim be the interim - (the member at 2 of xtuple).

[3] Produce the interim.

End.


To dedicate mass-product to xprop in B-props:

[1] If the xprop is a proposition in the B-props,
let the product-sum of the xprop be
   (the product-sum of the xprop) + the mass-product,
otherwise,
let the product-sum of the xprop be the mass-product
and assert the xprop is a proposition in the B-props.

End.


To generate tuple-representation from B-props:

Private combined-assignment.

[1] Let the combined-assignment be <>.

[2] For each proposition (B) in the B-props,
let the combined-assignment be the concatenation
   of (the combined-assignment) with
    <<B, the normalized-mass of B>>
and deny (the product-sum of B) is a product-sum of B
and deny (the normalized-mass of B) is a normalized-mass of B.

[3] Forget about the B-props.

[4] Produce the combined-assignment.

End.
```

```
[: K-CONF-MASTER  Created 27-Jan-83 13:05:52, edit by DILLARD :]

To generate combined-assignment of confidence-set:

Private full-list, full-length, gen-flag, special, general,
  short-list, short-length, unlike.

[1] Let the full-list be (the neg-prop-assignments in the confidence-set).

[2] Let the full-length be (the length of (the full-list)).

[3] If (the general-assignments in the confidence-set) = <>,
let the gen-flag be none,
otherwise,
let the gen-flag be yes
and let the general be (the general-combination of
   (the general-assignments in the confidence-set)).

[4] Let the special be (the special-combination of
   (the special-assignments in the confidence-set)).

[5] If the full-length = 0,
(if the gen-flag = none,
produce the special,
otherwise,
produce the special-gen-combination of (the special)
   with the general).

[6] If the full-length = 1,
(if the gen-flag = none,
produce the neg-special-combination of
   (the member at 1 of the full-list) with the special,
otherwise,
produce the special-gen-combination of (the special)
   with (the neg-gen-combination of
      (the member at 1 of the full-list) with the general)).

[7] Let the short-list be
   (the likes-combined-version of the full-list).

[8] Let the short-length be (the length of the short-list).

[9] If the short-length = 1,
(if the gen-flag = none,
produce the neg-special-combination of
   (the member at 1 of the short-list) with the special,
otherwise,
produce the special-gen-combination of (the special)
   with (the neg-gen-combination of
      (the member at 1 of the short-list) with the general)).
```

```
[10] Let the unlike be (the unlike-negs-combination of
   (the member at 1 of the short-list)
      with (the member at 2 of the short-list)).

[11] If the short-length > 2,
for each integer (i) from 3 to the short-length,
let the unlike be the neg-gen-combination of
   (the member at i of the short-list) with the unlike.

[12] If the gen-flag = none,
produce the special-gen-combination of (the special)
   with the general,
otherwise,
produce the special-gen-combination of (the special)
   with (the gen-gen-combination of (the general)
      with the unlike).

End.
```

```
[: K-CONF-NEGS  Created  8-Aug-83 11:46:01, edit by DILLARD :]

To generate likes-combined-version of neg-prop-assignments:

[Neg-prop-assignments must contain at least two assignments.
Output is a tuple containing neg-prop assignments, each for
a different proposition.]

Private resultant, prop.

[1] Let the resultant be <>.

[2] For each integer (i) from 1 to
      (the length of the neg-prop-assignments),
if there is a member (pair) at i of the neg-prop-assignments,
let the prop be (the member at 1 of pair)
and if the prop does occur,
let (the assignments for the prop) be the concatenation of
   (the assignments for the prop) with <pair>,
otherwise,
assert the prop is a proposition
   which does occur
and let (the assignments for the prop) be <pair>.

[3] For each proposition (p) which does occur,
let the resultant be the concatenation of (the resultant)
   with <the like-negs-combination of (the assignments for p)>
and deny p is a proposition
and p does occur
and (the assignments for p) is an assignments for p.

[4] Produce the resultant.

End.


To generate like-negs-combination of like-prop-negs:

[Like-prop-negs is a tuple containing neg-prop assignments, all
for the same proposition.  Output is a combined assignment in
neg-prop format.]

Private resultant, no-of-pairs, prop,
   next-tuple, next-mass.

[1] Let the resultant be the member at 1 of the like-prop-negs.

[2] Let the no-of-pairs be the length of the like-prop-negs.
```

```
[3] If the no-of-pairs = 1,
produce the resultant.

[4] Let the prop be the member at 1 of
   (the member at 1 of the like-prop-negs).

[5] For each integer (i) from 2 to the no-of-pairs,
let the next-tuple be the member at i of the like-prop-negs
and let the next-mass be the member at 2 of the next-tuple
and let the resultant be <the prop, 1 -
  (1 - (the member at 2 of the resultant)) * (1 - the next-mass)>.

[6] Produce the resultant.

End.


To generate unlike-negs-combination of C-assign with D-assign:

[Combines two neg-prop assignments for unlike propositions.
Output is a combined assignment in general format.]

Private Cnegprop, Cmass, Dnegprop, Dmass.

[1] Let the Cnegprop be the member at 1 of the C-assign.

[2] Let the Cmass be the member at 2 of the C-assign.

[3] Let the Dnegprop be the member at 1 of the D-assign.

[4] Let the Dmass be the member at 2 of the D-assign.

[5] Produce <<the logical-negation of the Cnegprop,
              the Cmass * (1 - the Dmass)>,
        <the logical-negation of the Dnegprop,
              the Dmass * (1 - the Cmass)>,
   <the logical-negation of <the Cnegprop, the Dnegprop>,
              the Cmass * the Dmass>>.

End.
```

```
[: K-CONF-SPECIAL-NEG  Created  1-Feb-83 11:59:23, edit by DILLARD :]

To generate special-combination of mass-vectors:

[Output is a combined mass assignment in the special format.]

Private resultant.

[1] Let the resultant be
   (the member at 1 of the mass-vectors).

[2] For each integer (i) from 2 to
    (the length of the mass-vectors),
let the resultant be the spec-spec-combination of
   (the member at i of the mass-vectors) with the resultant.

[3] Produce the resultant.

End.


To generate spec-spec-combination of vector1 with vector2:

[Output is a combined mass assignment in the special format.]

Private  u1, u2, numerators, ith-mass1, ith-mass2,
   constant, resultant.

[1] Let the u1 be 1 - (the tuple-sum of the vector1).

[2] Let the u2 be 1 - (the tuple-sum of the vector2).

[3] Let the numerators be <>.

[4] For each integer (i) from 1 to (the length of the vector1),
let the ith-mass1 be the member at i of the vector1
and the ith-mass2 be the member at i of the vector2
and the numerators be the concatenation (of the numerators) with
   <the ith-mass1 * the ith-mass2 + the ith-mass1 * the u2
    + the U1 * the ith-mass2>.

[5] Let the constant be the u1 * the u2 +
   (the tuple-sum of the numerators).

[6] Let the resultant be <>.

[7] For each integer (i) from 1 to (the length of the vector1),
let the resultant be the concatenation (of the resultant) with
   <(the member at i of the numerators) / the constant>.

[8] Produce the resultant.

End.
```

```
To generate neg-special-combination of neg-single with special:

[Used if all neg pro- assignments are for same prop plus if
no general-assignments.  Output is a combined mass-assignment
in a format peculiar to this case.]

Private negprop, negmass, count, index, indexed-mass, constant,
    input-u, odd-mass, resultant.

[1] Let the negprop be the member at 1 of the neg-single.

[2] Let the negmass be the member at 2 of the neg-single.

[3] Let the count be 0.

[4] For each member (m) of the prop-set,
Let the count be the count + 1
and if m = the negprop,
let the index be the count.

[5] Let the indexed-mass be the member at (the index)
    of the special.

[6] Let the constant be 1 - the negmass * the indexed-mass.

[7] Let the input-u be 1 -
   (the tuple-sum of the special).

[8] let the odd-mass be the indexed-mass *
   (1 - the negmass) / the constant.

[9] Let the resultant be <>.

[10] For each integer (i) from 1 to (the length of the special),
if i = the index,
let the resultant be the concatenation of (the resultant)
   with <the odd-mass>,
otherwise,
let the resultant be the concatenation of (the resultant)
   with <(the member at i of the special) / the constant>.

[11] Produce <*ORIG-PROPS*, the resultant,
   <*NOT*, the index, the negprop>,
      (the input-u * the negmass) / the constant>.

End.
```

```
[: K-CONF-NEG-GEN  Created 27-Jan-83 13:01:49, edit by DILLARD :]

To generate neg-gen-combination of C-assign with D-assign:

Private B-props, dead-mass, cnegprop, cmass.

[1] Let the B-props be B-props.

[2] Let the dead-mass be 0.

[3] Let the Cnegprop be the member at 1 of the C-assign.

[4] Let the Cmass be the member at 2 of the C-assign.

[5] For each member (D-tuple) of the D-assign.

go process-neg-gen-term (from the member at 1 of D-tuple)
   (under the member at 2 of D-tuple) (over the Cmass)
      (with the Cnegprop) (for the B-props).

[6] Go dedicate (the Cmass * (the theta-mass of the D-assign))
   (to (the logical-negation of the Cnegprop)) in the B-props.

[7] If the dead-mass = 0,
for each proposition (B) in the B-props,
let the normalized-mass of B be
   the product-sum of B,
otherwise,
for each proposition (B)in the B-props,
let the normalized-mass of B be
   (the product-sum of B) / (1 - the dead-mass).

[8] Produce the tuple-representation from the B-props.

End.


To process-neg-gen-term from Dprop
   under Dmass over Cmass with Cnegprop for B-props:

Private  Bprop, mass-product, intersect-flag.

[1] If the Dprop = <the Cnegprop>,
let the dead-mass be the dead-mass + the Cmass * the Dmass
and go dedicate ((1 - the Cmass) * the Dmass) (to the Dprop)
   in the B-props
and return.

[2] Let the Bprop be <>.

[3] Let the intersect-flag be non.
```

```
[4] For each member (D) of the Dprop,
if B = the Cnegprop,
let the intersect-flag be union,
otherwise,
let the Bprop be the concatenation of (the Bprop) with <B>.

[5] If the intersect-flag = non,
go dedicate (the Dmass) (to the Dprop) in the B-props,
otherwise,
go dedicate ((1 - the Cmass) * the Dmass)
   (to the Dprop) in the B-props
and go dedicate (the Cmass * the Dmass)
   (to the Bprop) in the B-props.

End.
```

```
[: K-CONF-SPEC-GEN  Created  1-Feb-83 12:21:09, edit by DILLARD :]

To generate special-gen-combination of special with general:

[Output is a combined mass assignment in the general format.]

Private B-props, index, indexed-mass, G-uncertainty, mass-product,
   dead-mass, special-uncertainty.

[1] Let the B-props be B-props.

[2] Let the index be 0.

[3] Let the dead-mass be 0.

[4] Let the G-uncertainty be the theta-mass of the general.

[5] For each member (prop) of the prop-set,
let the index be the index + 1
and let the indexed-mass be the member at (the index) of the special
and go dedicate (the indexed-mass * the G-uncertainty)
   (to <prop>) in the B-props
and for each member (Gpair) of the general,
let the mass-product be
     the indexed-mass * (the member at 2 of Gpair)
and if there is a member (m) (of (the member at 1 of Gpair))
   such that m = prop,
go dedicate (the mass-product) (to <prop>) in the B-props,
otherwise,
let the dead-mass be the dead-mass + the mass-product.

[6] Let the special-uncertainty be
      (1 - (the tuple-sum of the special)).

[7] For each member (Gpair) of the general,
go dedicate ((the member at 2 of Gpair) * the special-uncertainty)
   to (the member at 1 of Gpair) in the B-props.

[8] If the dead-mass = 0,
for each proposition (B) in the B-props,
let the normalized-mass of B be
   the product-sum of B,
otherwise,
for each proposition (B) in the B-props,
let the normalized-mass of B be
   (the product-sum of B) / (1 - the dead-mass).

[9] Produce the tuple-representation from the B-props.

End.
```

```
[: K-CONF-GENERAL  Created  1-Feb-83 09:39:35, edit by DILLARD :]

To generate general-combination of gen-assignments:

[Assumes gen-assignments has at least one member.]

Private glength, resultant.

[1] Let the glength be (the length of the gen-assignments).

[2] If the glength = 1,
produce the member at 1 of the gen-assignments.

[3] Let the resultant be (the member at 1 of the gen-assignments).

[4] For each integer (i) from 2 to the glength,
let the resultant be
   (the gen-gen-combination of (the resultant)
      with (the member at i of the gen-assignments)).

[5] Produce the resultant.

End.


To generate gen-gen-combination of C-assign with D-assign:

Private B-props.

[1] Let the B-props be B-props.

[2] Let the dead-mass be 0.

[3] For each member (C-tuple) of the C-assign,
for each member (D-tuple) of the D-assign,
go process-term from C-tuple (with D-tuple) (for the B-props).

[4] Go process-theta-terms (from the C-assign) (with
   (the theta-mass of the D-assign)) (for the B-props).

[5] Go process-theta-terms (from the D-assign) (with
   (the theta-mass of the C-assign)) (for the B-props).

[6] If the dead-mass = 0,
for each proposition (B) in the B-props,
let the normalized-mass of B be
   the product-sum of B,
otherwise,
for each proposition (B) in the B-props,
let the normalized-mass of B be
   (the product-sum of B) / (1 - the dead-mass).

[7] Produce the tuple-representation from the B-props.
```

```
End.

To process-term from C-tuple with D-tuple for B-props:

Private Cprop, Cmass, Dprop, Dmass, Bprop, mass-product.

[1] Let the Cprop be the member at 1 of the C-tuple.

[2] Let the Cmass be the member at 2 of the C-tuple.

[3] Let the Dprop be the member at 1 of the D-tuple.

[4] Let the Dmass be the member at 2 of the D-tuple.

[5] Let the Bprop be <>.

[6] For each member (C) of the Cprop,
for each member (D) of the Dprop,
if C = D,
let the Bprop be the concatenation of
   (the Bprop) with <C>.

[7] Let the mass-product be the Cmass * the Dmass.

[8] If the Bprop = <>,
let the dead-mass be the dead-mass + the mass-product,
otherwise,
go dedicate (the mass-product) (to the Bprop) in the B-props.

End.

To process-theta-terms from xassign with ytheta-mass for B-props:

[1] For each member (xtuple) of the xassign,
go dedicate ((the member at 2 of xtuple) * the ytheta-mass)
   to (the member at 1 of xtuple) in the B-props.

End.
```

```
[: K-CONF-OUTPUT  Created 23-May-83 13:18:12, edit by DILLARD :]

To give_type-confidences for track:

Private conf-set.

[1] If there is a platform (p) of the track
and there is a type (ty) of p,
send {return,
"The platform type is ", ty, ".", return}
and return.

[2] Let the conf-set be the type-conf-set of the track.

[3] If the general-assignments in the conf-set = <>
and the neg-prop-assignments in the conf-set = <>,
(if the length of (the special-assignments in the conf-set) = 1,
send {return,
"There has been no useful evidence yet concerning that track.
The following masses are based on a priori probabilities.", return}
and go list_type-confidences over (the special-pro-cons from
   (the member at 1 (of the special-assignments in the conf-set)))
and return,
otherwise,
go list_type-confidences over
   (the special-pro-cons from
      (the special-combination of
         (the special-assignments in the conf-set)))
and go offer_type-history for the track)
and return.

[4] Go list_type-confidences over (the pro-cons from
   (the combined-assignment of the conf-set))
and go offer_type-history for the track.

End.


To list_type-confidences over pro-con-pairs:

[1] Send {return, "   Platform Types:      Pro-Con Masses",
return, return}.

[2] Go list_confidences over (the pro-con-pairs)
      for the type-categories.

End.
```

```
To give_correl-confidences for track:

[Note that prop-set is not locally bound.]

Private conf-set.

[1] Let the conf-set be (the correl-conf-set of the track).

[2] Let the prop-set be (the prop-set of the conf-set).

[3] If the length of the prop-set = 1,
send {return,
"There are no candidate correlations for that track.", return}
and return.

[4] If the length of (the special-assignments in the conf-set) = 1,
go use_speed&course to the track
and if there is an initial-range of the track,
go compare_plat-sizes with the track.

[5] If the general-assignments in the conf-set = <>
and the neg-prop-assignments in the conf-set = <>,
go list_correl-confidences over
   (the special-pro-cons from
      (the special-combination of
         (the special-assignments in the conf-set)))
and go offer_correl-data for the track
and return.

[6] Go list_correl-confidences over (the pro-cons from
   (the combined-assignment of the conf-set))
and go offer_correl-data for the track.

End.


To list_correl-confidences over pro-con-pairs:

[1] Send {return, "   Candidate platforms:      Pro-Con Masses",
return, return}.

[2] Go list_confidences over (the pro-con-pairs)
      for the prop-set.

End.


To list_confidences over pro-con-pairs for prop-choices:

Private no-of-choices, max-measure, ith-pair, ith-prop, measure,
   most-likely.
```

```
[1] Let the no-of-choices be the length of the prop-choices.

[2] Let the max-measure be -1.

[3] For each integer (i) from 1 to the no-of-choices,
let the ith-pair be the member at i of the pro-con-pairs
and let the ith-prop be the member at i of the prop-choices
and the measure be ((the member at 1 of the ith-pair)
    - (the member at 2 of the ith-pair))
and (if the measure = the max-measure,
let the most-likely be the concatenation of
   (the most-likely) with <the ith-prop>)
and (if the measure > the max-measure,
let the max-measure be the measure
and the most-likely be <the ith-prop>)
and send (the ith-prop, ":     ", the ith-pair, return).

[4] Send (return, "The most likely ")
and if (the length of the most-likely) = 1,
send ("is ", the member at 1 of the most-likely, "."),
otherwise,
send ("are ", the most-likely, ".  (They tied.)").

[5] Send (return, return).

End.


To generate special-pro-cons from masses:

[From a mass assignment in the special format.]

Private pro-con-pairs,  ith-mass.

[1] Let the pro-con-pairs be <>.

[2] For each integer (i) from 1 to (the length of the masses),
let the ith-mass be the member at i of the masses
and the pro-con-pairs be the concatenation
   (of the pro-con-pairs) with <<the ith-mass,
    (the tuple-sum of the masses)  - the ith-mass>>.

[3] Produce the pro-con-pairs.

End.


To generate pro-cons from gen-assignment:

Private pro, con, pro-con-tuple, B.

[1] If the member at 1 of the gen-assignment = *ORIG-PROPS*,
produce the special-neg-pro-cons from the gen-assignment
and return.
```

```
[2] Let the pro-con-tuple be <>.

[3] For each member (prop) of the prop-set,
let the pro be 0
and the con be 0
and (for each member (pair) of the gen-assignment,
     let the B be (the member at 1 of pair)
     and if there is a member (m) of (the B) such that m = prop,
       (if the B = <prop>,
        let the pro be the member at 2 of pair),
     otherwise,
       let the con be the con + (the member at 2 of pair))
and let the pro-con-tuple be the concatenation of
   (the pro-con-tuple) with <<the pro, the con>>.

[4] Produce the pro-con-tuple.

End.


To generate special-neg-pro-cons from mixed-assignment:

[The mixed-assignment is in an peculiar format produced by
combining a special assignment with a neg-prop assignment.]

Private pro-con-tuple, special, negmass,
   negprop-index, uncertainty, pro, con.

[1] Let the pro-con-tuple be <>.

[2] Let the special be (the member at 2 of the mixed-assignment).

[3] Let the negmass be (the member at 4 of the mixed-assignment).

[4] Let the negprop-index be (the member at 2 of
   (the member at 3 of the mixed-assignment)).

[5] Let the uncertainty be
   1 - (the tuple-sum of the special) - the negmass.

[6] For each integer (i) from 1 to (the length of the special),
let the pro be the member at i of the special
and the con be 1 - the uncertainty - the pro
and if i = the negprop-index,
let the pro-con-tuple be the concatenation of
   (the pro-con-tuple) with <<the pro, the con>>,
otherwise,
let the pro-con-tuple be the concatenation of
   (the pro-con-tuple) with <<the pro, the con - the negmass>>.

[7] Produce the pro-con-tuple.

End.
```

```
[: K-HISTORY  Created 18-May-83 08:04:20, edit by DILLARD :]

To offer_type-history for track:

[1] Send {return,
"Would you like to see the history for this? (y or n)", return}
and read for 240 seconds {1 line (bind response)}

and if the name {response} = y
or the name {response} = Y,
go list_type-rules for the track
and return,
otherwise,
if the name {response} = n
or the name {response} = N,
return.

[2] Send {return,
"Didn't catch that answer.  y or n?", return}
and read for 240 seconds {1 line (bind response)}
and if the name {response} = y
or the name {response} = Y,
go list_type-rules for the track.

End.


To list_type-rules for track:

[1] Send {return,
"First, a mass assignment was made based on the likelihood
of a ship of each type being in this region; over the
eighteen ship types, it is <.003, .013, .04, .03, .04, .02,
.04, .06, .02, .015, .012, .09, .03, .01, .13, .07, .02, .007>.",
return}.

[2] Send {return,
"The ship-type rules that fired for ", the track,
" and their respective probability mass assignments are:",
return}.

[3] For each fired-rule (fr) of the track,
if fr is a ship-type-rule,
send {return, fr,
" Rule:  ", the pretty-assignment of fr to the track, return}.

End.
```

```
To generate pretty-assignment of rule to track:

[1] If there is a pretty-assignment (m) of the rule
produce m.

[2] If the rule = Speed,
produce the mass-vector_derived from
   <speed, the initial-speed of the track>.

[3] If the rule = Radar-Popup-Range,
produce the mass-vector_derived from
   <range, the initial-range of the track>.

[4] Produce unknown.

End.


[rule 1] Assert Course-Changed is a ship-type-rule
and " .6 to NOT merchant "
    is a pretty-assignment of Course-Changed.

[rule 2] Assert Square-Tie-Radar is a ship-type-rule
and " .5 to fast-attack/patrol-craft OR small-fighting-ship "
    is a pretty-assignment of Square-Tie-Radar.

[rule 3] Assert Sub-Surfacing is a ship-type-rule
and " .8 to submarine "
    is a pretty-assignment of Sub-Surfacing.

[rule 4] Assert Outside-All-Lanes is a ship-type-rule
and " .3 to NOT merchant "
    is a pretty-assignment of Outside-All-Lanes.

[rule 5] Assert Not-Known-Hostile is a ship-type-rule
and " .4 to merchant OR fishing OR other-commer/private "
    is a pretty-assignment of Not-Known-Hostile.

[rule 6] Assert Belated-Not-Known-Hostile is a ship-type-rule
and " .4 to merchant OR fishing OR other-commer/private "
    is a pretty-assignment of Belated-Not-Known-Hostile.

[rule 7] Assert Speed is a ship-type-rule.

[rule 8] Assert Radar-Popup-Range is a ship-type-rule.

To offer_correl-data for track:

[1] Send {return,
"Would you like to see individual contributions? (y or n)", return}
and read for 240 seconds {1 line (bind response)}
```

```
and if the name {response} = y
or the name {response} = Y,
go list_correl-contributions for the track
and return,
otherwise,
if the name {response} = n
or the name {response} = N,
return.

[2] Send {return,
"Didn't catch that answer.  y or n?", return}
and read for 240 seconds {1 line (bind response)}
and if the name {response} = y
or the name {response} = Y,
go list_correl-contributions for the track.

End.


To list_correl-contributions for track:

[Variable "prop-set" was bound in first listing.]

Private conf-set, specials.

[1] Let the conf-set be the correl-conf-set of the track.

[2] Let the specials be the special-assignments in the conf-set.

[3] Send {return,
"For the respective candidates ", the prop-set, ",",
"the following probability mass assignments were made",
return}.

[4] Send{return,
"   COVERAGE:  ", the member at 1 of the specials, return,
"   SPEED:     ", the member at 2 of the specials, return,
"   COURSE:    ", the member at 3 of the specials, return,
"   SIZE:      ", the member at 4 of the specials, return,
return, "Would you like to see underlying data, (y or n)",
return}
and read for 240 seconds {1 line (bind response)}

and if the name {response} = y
or the name {response} = Y,
go present_correl-data for (the conf-set) of the track
and return,
otherwise,
if the name {response} = n
or the name {response} = N,
return.

[5] Send {return,
"Didn't catch that answer.  y or n?", return}
```

```
and read for 240 seconds {1 line (bind response)}
and if the name {response} = y
or the name {response} = Y,
go present_correl-data for (the conf-set) of the track.

End.


To present_correl-data for conf-set of new-track:

Private xlength, speeds, courses, det-range, plat,
    last-rpt, xtype, xspeed, xcourse, xdet-range.

[1] Let the xlength be (the length of the prop-set) - 1.

[2] Let the courses be the average-courses of the conf-set.

[3] Let the speeds be the minimum-speeds of the conf-set.

[4] If there is an initial-range (ir) of the new-track,
let the det-range be ir,
otherwise,
let the det-range be unknown.

[5] For each integer (i) from 1 to the xlength,
let the plat be the member at i of the prop-set
and the xtrack be the track of the plat
and the last-rpt be the last-report of the xtrack
and (if there is a type (ty) of the plat,
let the xtype be ty,
otherwise,
let the xtype be unknown)
and (if there is a speed (sp) of the last-rpt,
let the xspeed be sp,
otherwise,
let the xspeed be unknown)
and (if there is a course (cs) of the last-rpt,
let the xcourse be cs,
otherwise,
let the xcourse be unknown)
and (if there is an initial-range (ir) of the xtrack,
let the xdet-range  be ir,
otherwise,
let the xdet-range be unknown)

and send {return, the plat, return,
"   Type:  ", the xtype, return,
"   Last spd:  ", the xspeed, "     Min spd to contact:  ",
the member at i of the speeds, return,
"   Last crs:  ", the xcourse, "     Avg crs to contact:  ",
the member at i of the courses, return}
and if the xdet-range ~= unknown,
send {
"   Initial detection range:  ", the xdet-range, return,
"      (New contact's i. ", the det-range, ")", return}.

End.
```

APPENDIX B

TYPESCRIPTS OF EXPERIMENTS

CONTENTS

B.1 Ship-Type Scenario: Report Processing

```
                    Th
[ Rosie     Version 1.3  17-Aug-83  07:22:48 ]

<2> load k-load.
To HELP
To START_CORREL-SCENARIO
To NEW-REPORT
To PROCESS_REPORT with REPORT
To generate TRACK of PLAT
To PRINT_AN_OUTLINE of RPT
To generate SUBTENDED_ANGLE of LATLONS
To generate ABS_VALUE of NUMBERC
To generate TIME_DIFFERENCE of TIMES
To generate TIME_DELTA of REPORTS
To generate DISTANCE of REPORTS
To generate MIN_SPEED of REPORTS
To generate DIRECTION of LATLONS
To generate AVERAGE-COURSE of TWO-REPORTS
To generate COURSE_DIFFERENCE of COURSES
To generate RANGE-TO-LANE-CENTER of POSITION to LANE
To generate REPORT_COUNT of TRACK
To generate FIRST-REPORT of TRACK
To generate LAST-REPORT of TRACK
To generate PLATFORM_TYPE of TRACK
To generate REACHABLES of TRACK
To decide SHIP1 is described_unlike SHIP2
To decide TRACK1 is not_platform-unlike TRACK2
To generate DESCRIPTION_SIMILARITY of SHIP1 to SHIP2
To decide SHIP is not_fully_identified
To generate MAX-SPEED of PLATFORM
To generate CORREL-CONF-SET of TRACK
To generate NORMAL-DENSITY for SIGMA of ARG with MEAN
To ASSIGN-TYPE-CONFIDENCES for TRACK of MASSES-OR-DATA to TYPE-CHOICES
To ASSIGN-CORREL-CONFIDENCES for TRACK of MASSES to PLAT-CHOICES
To ASSIGN-CONFIDENCES for CONFIDENCE-SET of MASSES to PROP-CHOICES
To generate TYPE-CONF-SET of TRACK
To generate MASS-VECTOR_DERIVED from SENSOR-MEASUREMENT
To generate SPECIAL-ASSIGNMENTS in CONF-SET
To generate NEG-PROP-ASSIGNMENTS in CONF-SET
To generate GENERAL-ASSIGNMENTS in CONF-SET
To generate SPECIAL-CONVERSION for PROP-CHOICES from MASSES
To generate GENERAL-CONVERSION for PROP-CHOICES from MASSES
To generate LOGICAL-NEGATION of NEG-PROP
To generate TUPLE-SUM of TUPLEX
To generate THETA-MASS of GEN-ASSIGNMENT
To DEDICATE MASS-PRODUCT in B-PROPS to XPROP
To generate TUPLE-REPRESENTATION from B-PROPS
```

```
For help, type:  help.

Would you prefer to see Scenario 1 (emphasizes platform type)
or Scenario 2 (emphasizes correlation)?

PLEASE TYPE THE INTEGER 1 OR 2
1
To RUN_TSA-RULES on CURRENT-REPORT
To TELL about ARG
To OFFER_TYPE-HISTORY for TRACK
To LIST_TYPE-RULES for TRACK
To generate PRETTY-ASSIGNMENT of RULE to TRACK
To RUN_CONTACT-ASSOC_RULES of CURRENT-TRACK on CURRENT-REPORT

To receive the first report, type:    new-report.
<3> new-report.

REPORT #1
   Platform:  SHIP1
   Track:     TRACK #1


 ... running-rules on report ...


Not-Known-Hostile Rule fires:
The position of the first-report of TRACK #1 is not
within reach of any platform identified as hostile.

Radar-Popup-Range Rule fires for TRACK #1.  (Range = 11.3)

Outside-All-Lanes Rule fires:
The position of REPORT #1 of TRACK #1 is outside all merchant lanes.


To receive the next report, type:  new-report.

<4> help.

For an explanation of a rule, type:
       tell about [rule-name].

To see confidence contributions concerning the platform type
of a track, type:  list_type-rules for track #[integer].

To see confidence contributions concerning contact correlation,
type:  list_correl-contributions for track #[integer].

To see evidential conclusions about platform-type for a
track, type:  give_type-confidences for track #[integer].
```

```
To see evidential conclusions about contact-correlation for a
track, type:  give_correl-confidences for track #[integer].
<5> tell about outside-all-lanes.

If the sighting is outside of all merchant lanes, then the
ship might not be a merchant. (.3).

<6> tell about radar-popup-range.

The initial radar detection range is indicative of the
platform's size and contributes confidence weights over all
platform types.

<7> track #1?
[ TRACK #1 ]
    TRACK #1 is a track.
    TRACK #1 is a track of SHIP1.
    SHIP1 is a platform of TRACK #1.
    REPORT #1 is a report in TRACK #1.
    TYPE-CONF-SET #1 is a type-conf-set of TRACK #1.
    NOT-KNOWN-HOSTILE is a fired-rule of TRACK #1.
    RADAR-POPUP-RANGE is a fired-rule of TRACK #1.
    OUTSIDE-ALL-LANES is a fired-rule of TRACK #1.
    11.3 is an initial-range of TRACK #1.

<8> report #1?
[ REPORT #1 ]
    REPORT #1 is a report.
    REPORT #1 is a report in TRACK #1.
    261110 is a time of REPORT #1.
    -7.3 is a latitude of REPORT #1.
    71.729 is a longitude of REPORT #1.
    11.3 is a range of REPORT #1.
    RADAR is a sensor of REPORT #1.
    90 is a course of REPORT #1.
    REPORT #1 is a radar-popup.

<9> new-report.

REPORT #2
   Platform:  SHIP1
   Track:     TRACK #1


 ... running-rules on report ...


Speed Rule fires for TRACK #1.   (Speed = 31)
```

```
To receive the next report, type:  new-report.

<10> tell about speed.

The speed measurement contributes confidence weights over all
platform types.

<11> new-report.

REPORT #3
   Platform:  SHIP1
   Track:     TRACK #1


 ... running-rules on report ...


Course-Changed Rule fires for TRACK #1:
 REPORT #3 course:  30
 REPORT #1 course:  90


To receive the next report, type:  new-report.

<12> tell about course-changed.

If the course has changed significantly, then the ship
probably isn't a merchant (.6).

<13> new-report.

REPORT #4
   Platform:  SHIP1
   Track:     TRACK #1


 ... running-rules on report ...


Square-Tie-Radar Rule fires for TRACK #1.


To receive the next report, type:  new-report.

<14> tell about square-tie-radar.

If the intercepted signal is a square tie radar, the contact is
likely to be a fast-attack/patrol-craft or a small-fighting-ship
(.55).

<15> new-report.

No more reports
<16> dump as k-stored-type.
<17> logout.
```

B.2 Ship-Type Scenario: Combining of Mass Assignments

```
                TM
[ Rosie    Version 1.3  17-Aug-83  07:54:59 ]

<2> load k-load-conf.
To generate LOGICAL-NEGATION of NEG-PROP
To generate TUPLE-SUM of TUPLEX
To generate THETA-MASS of GEN-ASSIGNMENT
To DEDICATE MASS-PRODUCT in B-PROPS to XPROP
To generate TUPLE-REPRESENTATION from B-PROPS
To generate COMBINED-ASSIGNMENT of CONFIDENCE-SET
To generate LIKES-COMBINED-VERSION of NEG-PROP-ASSIGNMENTS
To generate LIKE-NEGS-COMBINATION of LIKE-PROP-NEGS
To generate UNLIKE-NEGS-COMBINATION of C-ASSIGN with D-ASSIGN
To generate SPECIAL-COMBINATION of MASS-VECTORS
To generate SPEC-SPEC-COMBINATION of VECTOR1 with VECTOR2
To generate NEG-SPECIAL-COMBINATION of NEG-SINGLE with SPECIAL
To generate NEG-GEN-COMBINATION of C-ASSIGN with D-ASSIGN
To PROCESS-NEG-GEN-TERM for B-PROPS from DPROP over CMASS under DMASS
with CNEGPROP
To generate SPECIAL-GEN-COMBINATION of SPECIAL with GENERAL
To generate GENERAL-COMBINATION of GEN-ASSIGNMENTS
To generate GEN-GEN-COMBINATION of C-ASSIGN with D-ASSIGN
To PROCESS-TERM for B-PROPS from C-TUPLE with D-TUPLE
To PROCESS-THETA-TERMS for B-PROPS from XASSIGN with YTHETA-MASS
To GIVE_TYPE-CONFIDENCES for TRACK
To LIST_TYPE-CONFIDENCES over PRO-CON-PAIRS
To GIVE_CORREL-CONFIDENCES for TRACK
To LIST_CORREL-CONFIDENCES over PRO-CON-PAIRS
To LIST_CONFIDENCES for PROP-CHOICES over PRO-CON-PAIRS
To generate SPECIAL-PRO-CONS from MASSES
To generate PRO-CONS from GEN-ASSIGNMENT
To generate SPECIAL-NEG-PRO-CONS from MIXED-ASSIGNMENT
To OFFER_TYPE-HISTORY for TRACK
To LIST_TYPE-RULES for TRACK
To generate PRETTY-ASSIGNMENT of RULE to TRACK
To OFFER_CORREL-DATA for TRACK
To LIST_CORREL-CONTRIBUTIONS for TRACK
To PRESENT_CORREL-DATA for CONF-SET of NEW-TRACK
To ASSIGN-TYPE-CONFIDENCES for TRACK of MASSES-OR-DATA to TYPE-CHOICES
To ASSIGN-CORREL-CONFIDENCES for TRACK of MASSES to PLAT-CHOICES
To ASSIGN-CONFIDENCES for CONFIDENCE-SET of MASSES to PROP-CHOICES
To generate TYPE-CONF-SET of TRACK
To generate MASS-VECTOR_DERIVED from SENSOR-MEASUREMENT
To generate SPECIAL-ASSIGNMENTS in CONF-SET
To generate NEG-PROP-ASSIGNMENTS in CONF-SET
To generate GENERAL-ASSIGNMENTS in CONF-SET
To generate SPECIAL-CONVERSION for PROP-CHOICES from MASSES
To generate GENERAL-CONVERSION for PROP-CHOICES from MASSES
```

```
<3> restore k-stored-type.
<4> give_type-confidences for track #1.

   Platform Types:      Pro-Con Masses

CARRIER:      <.02339607, .8893958>
CRUISER:      <.02652933, .8862625>
DESTROYER:      <.03498914, .8778027>
FRIGATE:      <.01433951, .8984524>
AMPHIBIOUS:      <.01191014, .9008817>
SUBMARINE:      <.01868374, .8941081>
SMALL-FIGHTING-SHIP:      <.072544, .7336602>
FAST-ATTACK/PATROL-CRAFT:      <.1971745, .6090297>
PATROL-CRAFT:      <.04979143, .8630004>
INTELL-COLLECTOR:      <.01390521, .8988867>
SURVEY/RESEARCH:      <.004793955, .9079979>
FLEET-AUXIL-MED/LRG:      <.02461761, .8881743>
FLEET-AUXIL-SMALL:      <.02385516, .8889367>
SMALL-BOAT:      <.0218556, .8909363>
MERCHANT:      <.04970593, .867737>
FISHING:      <.07986374, .7747894>
OTHER-COMMER/PRIVATE:      <.06205827, .7925949>
DEBRIS:      <.01805208, .8947398>

The most likely is FAST-ATTACK/PATROL-CRAFT.


Would you like to see the history for this? (y or n)
y

First, a mass assignment was made based on the likelihood
of a ship of each type being in this region; over the
eighteen ship types, it is <.003, .013, .04, .03, .04, .02,
.04, .06, .02, .015, .012, .09, .03, .01, .13, .07, .02, .007>.

The ship-type rules that fired for TRACK #1
 and their respective probability mass assignments are:

NOT-KNOWN-HOSTILE Rule:
 .4 to merchant OR fishing OR other-commer/private

RADAR-POPUP-RANGE Rule:
<0, 0, 0, 0, 0, .037, .045, .136, .136, .027, 0, 0, .045, .055,
.023, .077, .064, .055>

OUTSIDE-ALL-LANES Rule:   .3 to NOT merchant

SPEED Rule:
<.103, .103, .103, .029, .008, .009, .029, .074, .009, .008, .008,
.008, .008, .011, .034, .011, .045, 0>
```

```
COURSE-CHANGED Rule:    .6 to NOT merchant

SQUARE-TIE-RADAR Rule:
 .5 to fast-attack/patrol-craft OR small-fighting-ship
<5> logout.
```

B.3 Contact-Association Scenario: Report Processing

```
                TM
[ Rosie    Version 1.3  17-Aug-83  08:04:57 ]

<2> load k-load.
To HELP
To START_CORREL-SCENARIO
To NEW-REPORT
To PROCESS_REPORT with REPORT
To generate TRACK of PLAT
To PRINT_AN_OUTLINE of RPT
To generate SUBTENDED_ANGLE of LATLONS
To generate ABS_VALUE of NUMBERC
To generate TIME_DIFFERENCE of TIMES
To generate TIME_DELTA of REPORTS
To generate DISTANCE of REPORTS
To generate MIN_SPEED of REPORTS
To generate DIRECTION of LATLONS
To generate AVERAGE-COURSE of TWO-REPORTS
To generate COURSE_DIFFERENCE of COURSES
To generate RANGE-TO-LANE-CENTER of POSITION to LANE
To generate REPORT_COUNT of TRACK
To generate FIRST-REPORT of TRACK
To generate LAST-REPORT of TRACK
To generate PLATFORM_TYPE of TRACK
To generate REACHABLES of TRACK
To decide SHIP1 is described_unlike SHIP2
To decide TRACK1 is not_platform-unlike TRACK2
To generate DESCRIPTION_SIMILARITY of SHIP1 to SHIP2
To decide SHIP is not_fully_identified
To generate MAX-SPEED of PLATFORM
To generate CORREL-CONF-SET of TRACK
To generate NORMAL-DENSITY for SIGMA of ARG with MEAN
To ASSIGN-TYPE-CONFIDENCES for TRACK of MASSES-OR-DATA to TYPE-CHOICES
To ASSIGN-CORREL-CONFIDENCES for TRACK of MASSES to PLAT-CHOICES
To ASSIGN-CONFIDENCES for CONFIDENCE-SET of MASSES to PROP-CHOICES
To generate TYPE-CONF-SET of TRACK
To generate MASS-VECTOR_DERIVED from SENSOR-MEASUREMENT
To generate SPECIAL-ASSIGNMENTS in CONF-SET
To generate NEG-PROP-ASSIGNMENTS in CONF-SET
To generate GENERAL-ASSIGNMENTS in CONF-SET
To generate SPECIAL-CONVERSION for PROP-CHOICES from MASSES
To generate GENERAL-CONVERSION for PROP-CHOICES from MASSES
To generate LOGICAL-NEGATION of NEG-PROP
To generate TUPLE-SUM of TUPLEX
To generate THETA-MASS of GEN-ASSIGNMENT
To DEDICATE MASS-PRODUCT in B-PROPS to XPROP
To generate TUPLE-REPRESENTATION from B-PROPS
```

```
For help, type:  help.

Would you prefer to see Scenario 1 (emphasizes platform type)
or Scenario 2 (emphasizes correlation)?

PLEASE TYPE THE INTEGER 1 OR 2
2
To RUN_CONTACT-ASSOC_RULES of CURRENT-TRACK on CURRENT-REPORT
To USE_SPEED&COURSE to LATER-TRACK
To COMPARE_PLAT-SIZES with LATER-TRACK
To generate COURSE-SOUNDNESS for PLATFORM of TWO-POSIT-CRS
To generate SPEED-SOUNDNESS for PLATFORM of TWO-POSIT-SPD
To generate SIZE-SOUNDNESS for PLATFORM of INIT-RANGE
To generate TYPE-SIZE-SOUNDNESS of TYPE with RANGE
To generate NORMALIZED-SOUNDNESS-MEASURES from MEASURES
To RUN_TSA-RULES on CURRENT-REPORT

(Loading background data about earlier contacts and sightings.)
To OFFER_CORREL-DATA for TRACK
To LIST_CORREL-CONTRIBUTIONS for TRACK
To PRESENT_CORREL-DATA for CONF-SET of NEW-TRACK

(Receiving message about new contact.)

REPORT #6
   Platform:  SHIP1
   Track:     TRACK #6


A minor r le fires:
The posit.on of the first-report of TRACK #6 can be
reached from the position of the last-report of:
     TRACK #1
     TRACK #2
     TRACK #3
     TRACK #4
     TRACK #5


A minor rule fires:
A platform that could have reached TRACK #6 is hostile:
   AGI1.


A minor rule fires:
A platform that could have reached TRACK #6 is hostile:
   DESTROYER1.

<3> help.
```

```
For an explanation of a rule, type:
       tell about [rule-name].

To see confidence contributions concerning the platform type
of a track, type:  list_type-rules for track #[integer].

To see confidence contributions concerning contact correlation,
type:  list_correl-contributions for track #[integer].

To see evidential conclusions about platform-type for a
track, type:  give_type-confidences for track #[integer].

To see evidential conclusions about contact-correlation for a
track, type:  give_correl-confidences for track #[integer].
<4> list_correl-contributions for track #6.

For the respective candidates
<MERCHANT1, AGI1, DESTROYER1, MERCHANT2, MERCHANT3, NONE-OF-THE-ABOVE>,
the following probability mass assignments were made

   COVERAGE:  <0, 0, 0, 0, 0, .6>
   SPEED:     <0.0, 0.0, .2333333, 0.0, .4666667, 0.0>
   COURSE:    <0.0, .08402364, .08402364, .244539, .2751865, .01222724>
   SIZE:
<.1332379, .1438888, .08196076, .1853296, .1437162, .01186674>

Would you like to see underlying data, (y or n)
y

MERCHANT1
   Type:  MERCHANT
   Last spd:  20      Min spd to contact:  26.76568
   Last crs:  280      Avg crs to contact:  102.5702
   Initial detection range:  19
      (New contact's is 21.49)

AGI1
   Type:  AGI
   Last spd:  13      Min spd to contact:  22.9047
   Last crs:  230      Avg crs to contact:  137.5211

DESTROYER1
   Type:  DESTROYER
   Last spd:  33      Min spd to contact:  19.76664
   Last crs:  130      Avg crs to contact:  211.5286

MERCHANT2
   Type:  MERCHANT
   Last spd:  18      Min spd to contact:  28.82616
   Last crs:  103      Avg crs to contact:  104.5475
   Initial detection range:  22
      (New contact's is 21.49)
```

```
MERCHANT3
   Type:  MERCHANT
   Last spd:  28      Min spd to contact:  27.86344
   Last crs:  282.5      Avg crs to contact:  283.0192
<5> dump as k-stored-correl.
<6> logout.
```

B.4 Contact-Association: Combining of Mass Assignments

```
                    TM
          [ Rosie    Version 1.3  17-Aug-83  08:14:26 ]

<2> load k-load-conf.
To generate LOGICAL-NEGATION of NEG-PROP
To generate TUPLE-SUM of TUPLEX
To generate THETA-MASS of GEN-ASSIGNMENT
To DEDICATE MASS-PRODUCT in B-PROPS to XPROP
To generate TUPLE-REPRESENTATION from B-PROPS
To generate COMBINED-ASSIGNMENT of CONFIDENCE-SET
To generate LIKES-COMBINED-VERSION of NEG-PROP-ASSIGNMENTS
To generate LIKE-NEGS-COMBINATION of LIKE-PROP-NEGS
To generate UNLIKE-NEGS-COMBINATION of C-ASSIGN with D-ASSIGN
To generate SPECIAL-COMBINATION of MASS-VECTORS
To generate SPEC-SPEC-COMBINATION of VECTOR1 with VECTOR2
To generate NEG-SPECIAL-COMBINATION of NEG-SINGLE with SPECIAL
To generate NEG-GEN-COMBINATION of C-ASSIGN with D-ASSIGN
To PROCESS-NEG-GEN-TERM for B-PROPS from DPROP over CMASS under DMASS
with CNEGPROP
To generate SPECIAL-GEN-COMBINATION of SPECIAL with GENERAL
To generate GENERAL-COMBINATION of GEN-ASSIGNMENTS
To generate GEN-GEN-COMBINATION of C-ASSIGN with D-ASSIGN
To PROCESS-TERM for B-PROPS from C-TUPLE with D-TUPLE
To PROCESS-THETA-TERMS for B-PROPS from XASSIGN with YTHETA-MASS
To GIVE_TYPE-CONFIDENCES for TRACK
To LIST_TYPE-CONFIDENCES over PRO-CON-PAIRS
To GIVE_CORREL-CONFIDENCES for TRACK
To LIST_CORREL-CONFIDENCES over PRO-CON-PAIRS
To LIST_CONFIDENCES for PROP-CHOICES over PRO-CON-PAIRS
To generate SPECIAL-PRO-CONS from MASSES
To generate PRO-CONS from GEN-ASSIGNMENT
To generate SPECIAL-NEG-PRO-CONS from MIXED-ASSIGNMENT
To OFFER_TYPE-HISTORY for TRACK
To LIST_TYPE-RULES for TRACK
To generate PRETTY-ASSIGNMENT of RULE to TRACK
To OFFER_CORREL-DATA for TRACK
To LIST_CORREL-CONTRIBUTIONS for TRACK
To PRESENT_CORREL-DATA for CONF-SET of NEW-TRACK
To ASSIGN-TYPE-CONFIDENCES for TRACK of MASSES-OR-DATA to TYPE-CHOICES
To ASSIGN-CORREL-CONFIDENCES for TRACK of MASSES to PLAT-CHOICES
To ASSIGN-CONFIDENCES for CONFIDENCE-SET of MASSES to PROP-CHOICES
To generate TYPE-CONF-SET of TRACK
To generate MASS-VECTOR_DERIVED from SENSOR-MEASUREMENT
To generate SPECIAL-ASSIGNMENTS in CONF-SET
To generate NEG-PROP-ASSIGNMENTS in CONF-SET
To generate GENERAL-ASSIGNMENTS in CONF-SET
To generate SPECIAL-CONVERSION for PROP-CHOICES from MASSES
To generate GENERAL-CONVERSION for PROP-CHOICES from MASSES
```

```
<3> restore k-stored-correl.
<4> give_correl-confidences for track #6.

   Candidate platforms:      Pro-Con Masses

MERCHANT1:      <.03144517, .8977525>
AGI1:      <.06330025, .8658974>
DESTROYER1:      <.1343418, .7948559>
MERCHANT2:      <.1371054, .7920922>
MERCHANT3:      <.4423003, .4868974>
NONE-OF-THE-ABOVE:      <.1207047, .808493>

The most likely is MERCHANT3.


Would you like to see individual contributions? (y or n)
n
<5> logout.
```